C Programming

Manjunath R, Dennis Ritchie

manjunath5496@gmail.com

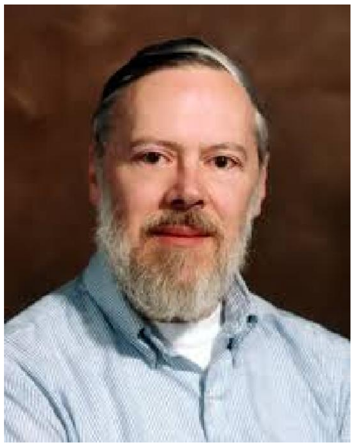
Abstract: A High Level Programming Language (which uses alphabets, digits, punctuations and some special symbols and cannot be executed directly without being converted into machine level language (the language which uses only 0 and 1))

[Manjunath R, Dennis Ritchie. **C Programming.** *Rep Opinion* 2018;10(1):43-196]. ISSN 1553-9873 (print); ISSN 2375-7205 (online). http://www.sciencepub.net/report. 9. doi:10.7537/marsroj100118.09.

Keywords: C Programming; Language; alphabet; digits; punctuation; symbol

C Programming

A High Level Programming Language (which uses alphabets, digits, punctuations and some special symbols and cannot be executed directly without being converted into machine level language (the language which uses only 0 and 1)) developed by a man named Dennis Ritchie in 1970s at Bell Telephone laboratories (now named AT & T Bell laboratories), Murray Hill, New Jersey to develop the UNIX operating system.



Using the two early programming languages -- Basic Combined Programming Language (BCPL) and BASIC (Beginner's All-purpose Symbolic Instruction Code) language Uses: used in the development of a) operating systems like LINUX, UNIX b) embedded systems like ATMs, printers. Most of the state-of-the-art software has been developed using C.

Advantages: relatively simple language, reliable (able to be trusted), easy to understand, easy to use, write, modify and debug and quick to learn.

MYSQL Database, LINUX Drivers, Text Editors are written in C.

C is called a structured programming language because it divides the problem into smaller modules called functions or procedures each of which handles a particular responsibility. Hence it is simple and easy to understand and well suited for small size implementation. However this is not restricted. A large size implementation is possible but complex design and full object oriented design cannot be implemented (because complex design concepts like Polymorphism and inheritance are not available in C).

A Simple C program basically comprises of the following parts:

Preprocessor Commands
Functions
Variables
Statements & Expressions
Comments

```
//
---------------------------------------------------------------------------------
/* My First C Program */ Comment
#include<stdio.h> // preprocessor command
int main () // Function where the program execution begins.
{
printf ("Hello,world!"); // statement
return 0;
}
------------------------------------------------------------------------------------ //
```

Process of C Program Execution: A C program:

```
#include<stdio.h>
int main ()
{
printf ("Hello,world!");
return 0;
}
```

is written using Text Editor, such as [Notepad (in case of Windows Operating System), vim or vi (in case of Linux Operating System)] and saved with [.C] Extension.

File Saved with [.C] extension is called Source Program or Source Code.

C Source code with [.C] Extension is sent to preprocessor first.

The preprocessor generates an expanded source code:

//

The contents of <stdio.h> would be pasted at the location of #include<stdio.h>

```
int main ()
{
printf ("Hello,world!");
return 0;
}
```

--- //

Expanded source code is given as input to compiler where the expanded source program is compiled (i.e., the program is entirely read and translated to instructions the computer can understand i.e., machine understandable / readable language i.e., to machine code sequence of 0s and 1s). If the C compiler finds any error during compilation, it provides information about the error to the programmer.

The programmer has to review code and re-edit the program. After re-editing program, Compiler again check for any error.

If program is error-free then it is sent to assembler (where the code is assembled and converted into object code. Now a simple.obj file is generated).

The object code is sent to linker (where the object code is linked with appropriate libraries). Then it is converted into a single executable code. A simple.exe file is generated.

The executable code is sent to loader (where the executable code is loaded into memory and then it is executed).

After execution, output:

```
Hello,world!
```

is displayed on the console screen.

C is case sensitive language: only lower case letters (or small letters) must be used.

Capital letters (or upper case letters) must be avoided to prevent the display of error on the screen

(For example: If the statement PRINTF ("Hello,world!"); is written instead of printf ("Hello,world!"); or INT MAIN () is written instead of int main (), compilation Error will be displayed on the console screen).

Parentheses () indicate a function and the word main indicate the name of the function. main () implies: main function

And if we forget to end each statement within the body of the main function with a semicolon (;), then the compilation Error will be displayed on the screen.

There should be no space between main and the parentheses ()

i.e., int main ()

and there should be no space inside the parentheses ()

i.e., int main ()

to prevent the display of compilation error on the screen.

As we know C is Platform dependent language. So the Operating system needs to know when the program execution ends.

So when there is value returns from the main function the Operating System get to know that the program execution is over.

int main () implies: main () should return integer value.

If the main function returns 0 to the operating system, then the program has completed execution successfully.

If the main function returns 1 to the operating system, then the program has not completed execution successfully.

The statement

#include<stdio.h>

tells the compiler to include the text from the file stdio.h (which is already present in the operating system) before it translates or compiles the program into a sequence of 0s and 1s. stdio means standard input output and stdio.h means standard input output header file (printf () & scanf () are not part of the C language, because there is no input or output defined in C language itself-- stdio.h comprises standard input output functions like scanf, printf etc.

and allows standard input /output operations -- note: scanf is an input function and printf is an output function (note: Letter f denote formatted) and it is included into the C program by writing the statement #include <stdio.h>).

If a program is written without the statement:

#include<stdio.h>, then the C compiler can't compile and a compilation error is displayed on the screen (because C compiler fails to recognize the functions such as printf () and scanf ()).

Note: We can also write #include "stdio.h" instead of #include <stdio.h> but some online compilers will flag error message. So the statement #include <stdio.h> is generally preferred and the statement #include "stdio.h" is generally ignored.

main () The program begins its execution with the function main () -- which is called the user defined function (because this function is defined by the user) - the main function -- the entry point of the program execution i.e., the point from where the execution of C program begins and the point at which the operating system passes control of the computer over to that program.

int main () {

} implies body of the main function within which the sequence of instructions in the form of statements i.e., the program is written and executed. The left curly brace

{

implies: the beginning of the main function

and the right curly brace

}

implies: the end of the main function.

return 0; implies the exit status of execution of the program i.e., at this point,

main function returns back the control of the computer to the operating system since

the execution is terminated at this point and once a return statement

i.e., return 0; is executed, no further instructions within the main function are executed

For example:

```
#include<stdio.h>
int main ()
{
printf ("Hello,world!");
return 0;
printf ("Hello,world!");
}
```

Output on the screen:

```
Hello,world!
```

; implies semicolon or statement terminator A program is a well-defined set of instructions and each well-defined instruction (in the form of a statement) is ended by a semicolon (which is C language punctuation -- like a period in English i.e., in an English paragraph each sentence is ended by a full stop which tells that one sentence ends and another begins, semicolon implies the end of one logical entity -- that one instruction (or statement) ends and another begins).

printf () output function of the C language which makes provision to print the output:

Hello,world!

on the screen. Parentheses () indicate a function and the word printf indicate the name of the function.

The text

Hello,world!

should be enclosed by the double quotation marks ("") and should be written within the printf function and this

printf function should be ended with the semicolon i.e.,

printf ("Hello,world!");

otherwise the compilation error will be displayed on the screen.

Program 1.1

C program to print the word "hello Bill Gates" on screen

```
#include<stdio.h>
int main ()
{
printf ("hello Bill Gates");
return 0;
}
```

The output on the screen:

```
hello Bill Gates
```

Program 1.2

C program to print

```
*
*****
*****
*****
*****
```

on screen

```
#include<stdio.h>
int main ()
{
printf ("\n        *      ");
printf ("\n ***** ");
printf ("\n ***** ");
printf ("\n ***** ");
printf ("\n ***** ");
return 0;
}
```

The output on the screen:

```
*
*****
*****
*****
*****
```

If new line sequence (\n) is not included in the above program then the output on the screen is:

```
*********************
```

In the above code, the new line character has the ASCII value of 10 which means it is a new line. The author has put the ASCII values table for different characters that can be used in the programs to solve real time problems.

Write a program to print the following outputs:

(a)

```
*
****
*******
****
*
```

(b)

```
****************
* *
* Hello World! *
* *
****************
```

(c)

```
Braces come in pairs!
Comments come in pairs!
All statements end with a semicolon!
Spaces are optional!
```

Must have a main function!

C is done mostly in lowercase. It's a case-sensitive language

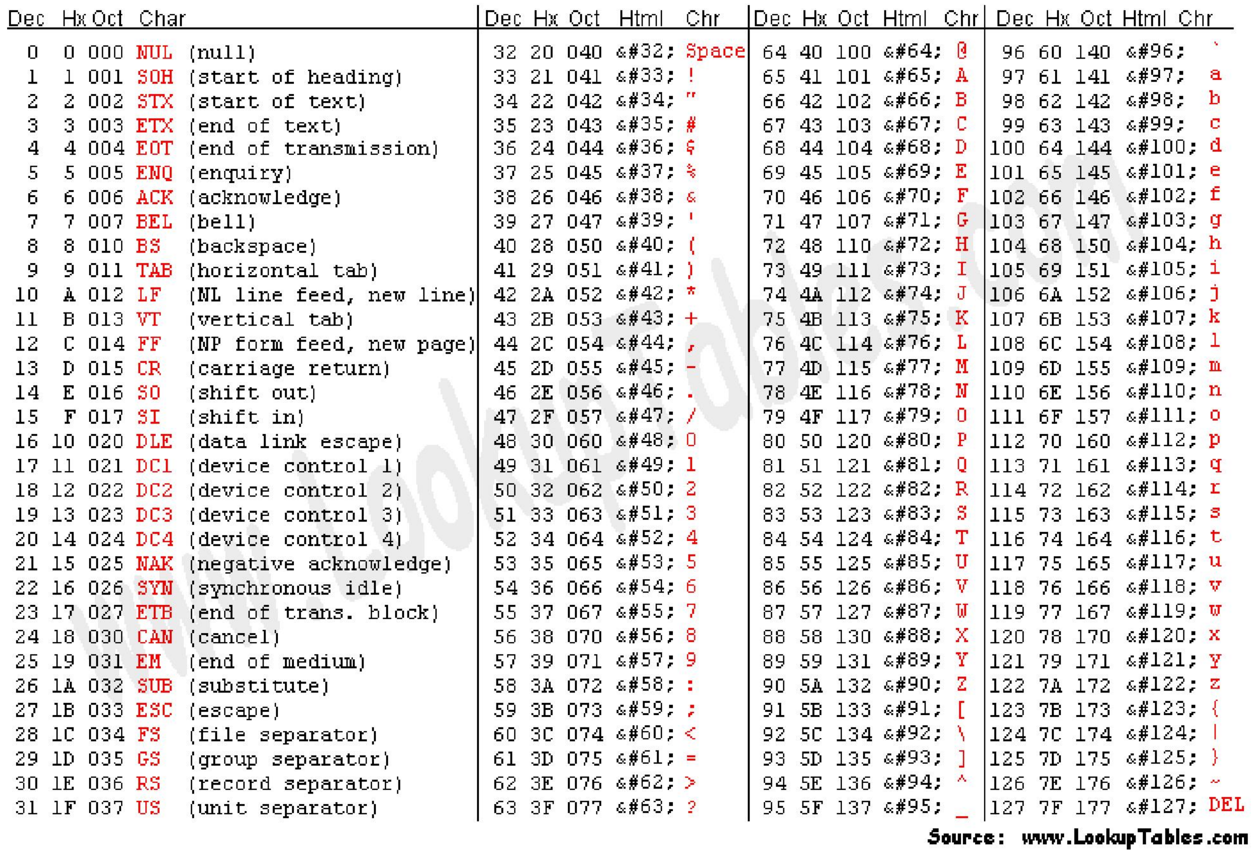
Answers:

(a)

```
#include<stdio.h>
int main ()
{
printf ("\n      *     ");
printf ("\n      **** ");
printf ("\n      ******* ");
printf ("\n      ****        ");
printf ("\n      *     ");
return 0;
}
```

(b)

```
#include<stdio.h>
int main ()
{
printf ("\n      ****************        ");
printf ("\n      * *    ");
printf ("\n      * Hello World! *      ");
printf ("\n      * *    ");
printf ("\n      ****************        ");
return 0;
}
```



Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT (end of transmission)	36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	VT (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	FF (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	CR (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	SO (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	SI (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	DLE (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	DC1 (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	DC2 (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	DC3 (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	DC4 (device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	NAK (negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	SYN (synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	ETB (end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN (cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	EM (end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	SUB (substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	ESC (escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
29	1D	035	GS (group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	US (unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

(c)

```
#include<stdio.h>
int main ()
{
printf ("\n Braces come in pairs!");
printf ("\n Comments come in pairs!");
printf ("\n All statements end with a semicolon!");
printf ("\n Spaces are optional!");
printf ("\n Must have a main function!");
printf ("\n C is done mostly in lowercase. It's a case-sensitive language");
return 0;
}
```

Program 1.3

C program to find the area of a circle

```
#include<stdio.h>
int main ()
{
int r, area;
r = 2;
area = 4 * 3.14 * r * r;
printf ("The area of the circle = %d", area);
return 0;
}
```

The output on the screen:

```
The area of the circle = 50
```

int means the data type (an attribute that tells what kind of data that value can own) is integer and the storage size of int data type is 2 or 4 or 8 byte.

Note:

A string, for example, is a data type that is used to categorize text and an integer is a data type used to categorize whole numbers / non fractional values.

We can't store decimal values using int data type.

If we use int data type to store decimal values, decimal values will be shortened and we will get only whole number. In this case, float data type can be used to store decimal values in a variable.

The statement

```
int r, area;
```

imply that we are creating the integer variables r, area.

Equal sign (" = ") implies storage operator.

The statements

```
r = 2;
area = 4 * 3.14 * r * r;
```

imply that we are storing the values to the created variables (i.e., we are storing the value 2 for r and 4 * 3.14 * r * r = 4 * 3.14 * 2 * 2 = 50 for area).

Comma in the statement

```
int r, area;
```

imply variable separator.

The statement

```
printf ("The area of the circle = %d", area);
```

make provision to print the output:

The area of the circle = 50

on the screen.

In the statement

```
printf ("The area of the circle = %d", area);
```

format string or format specifier %d indicates that the integer value to be displayed after the statement

The area of the circle =

enclosed by double quotes needs to be taken from a variable area.

The area of the circle is 50. 24 (for r = 2) but The area of the circle = 50 is displayed on the screen because data type int is used instead of float and format specifier %d is used instead of %f.

If float r, area; is used instead of int r, area; and

If the statement

```
printf ("The area of the circle = %f", area);
```

is written instead of

```
printf ("The area of the circle = %d", area);
```

i.e.,

```
#include<stdio.h>
int main ()
{
float r, area;
r = 2;
area = 4 * 3.14 * r * r;
printf ("The area of the circle = %f", area);
return 0;
}
```

Then the output on the screen:

```
The area of the circle = 50.24
```

float means the data type is float.

The statement

```
float r, area;
```

imply that we are creating the floating variables r, area.

(floating point variable means fractional variable or decimal number (for example: 1.5, 2.5, 3.5, 4.7 etc.) whereas integer means non-fractional variable or whole number (for example: 1, 2, 3, 4 etc.))

data type float is used instead of int (and format string %f is used instead of %d) because if the data type int is used instead of float then the result will not be clearly outputted i.e., instead of 50.24 the computer displays only 50.

If the statement

```
printf ("The area of the circle = %2f", area);
```

is written instead of the statement

```
printf ("The area of the circle = %f", area);
```

Then the output on the screen is:

```
The area of the circle =     50.24
```

i.e., the statement

printf ("The area of the circle = %f", area); yields the output:

```
The area of the circle = 50.24
```

whereas the statement

printf ("The area of the circle = %2f", area); yields the output:

```
The area of the circle =     50.24
```

If you want to supply the value for r through the key board, then the statement

```
r =2;
```

should be replaced by the statements

```
printf ("Enter any number:");
scanf ("%d",     &   r);
```

i.e., the program should be rewritten as:

```
#include<stdio.h>
int main ()
{
float r, area;
printf ("Enter any number:");
scanf ("%d",     &   r);
area = 4 * 3.14 * r * r;
printf ("The area of the circle = %f", area);
return 0;
}
```

The output on the screen:

```
Enter any number:
```

If you enter the number 2

The area of the circle = 50.24 will be outputted on the screen.

The statement

float r, area;

imply that we are creating the float variables r and area and these variables are stored in the computer memory and they are assigned an address to locate their position in the computer memory (like houses in a street are assigned an address to locate their position in the street).

The statement

printf ("Enter any number:");

make provision to print the text

Enter any number:

on the screen.

The statement

scanf ("%d", & r);

make provision to enter a number for r through the keyboard and store the number entered for r through the keyboard in the address of r in the computer memory. & symbol imply the address and & r imply the address of r in the computer memory.

Format string %d in the statement scanf ("%d", & r); tells the input function scanf to read the number entered through the keyboard (which is a integer) and since "%d" is followed by, & r --- %d tells the scanf function to read the integer entered through the keyboard for r and store it in the address of r in the computer memory (i.e., store the number in & r).

Note:

As told earlier: when you enter an integer for r through the keyboard, this integer will be stored in the computer memory. If you want to know the storage size of the integer in computer memory (i.e., space occupied by the entered integer in the computer memory), you need to appeal to the following program:

```
#include <stdio.h>
int main ()
{
int r;
r=10;
printf ("size of r = %d", sizeof (r));
return 0;
}
```

The output on the screen:

```
size of r = 4
```

i.e., integer entered for r i.e., 10 has occupied a space of 4 bytes in the computer memory.

Write a program to print the circumference of the circle (given r = 2.5)

Answer:

```
#include<stdio.h>
int main ()
{
float r, area;
r = 2.5;
circumference = 3.14 * r * r;
printf ("The circumference of the circle = %f", circumference);
return 0;
}
```

Write a program to print the area of the rectangle (given l = 2.5 and b = 3)

Answer:

```
#include<stdio.h>
int main ()
{
float l, b, area;
l = 2.5;
b = 3;
area = 1*b;
printf ("The area of the rectangle = %f", area);
return 0;
}
```

Format Specifiers in C

Data Type	Format Specifier
int	%d
float	%f or %e
char	%c
double	%lf or %le
long int	%ld

Program 1.3

C program to find the sum of two numbers

```
#include<stdio.h>
int main ()
{
int a, b, sum;
a=1;
b=2;
sum = a + b;
printf ("the sum of a and b = %d", sum);
return 0;
}
```

The output on the screen:

```
the sum of a and b = 3
```

If you want to assign the floating point values i.e., fractional numbers for a & b (i.e., 1.5 for a & 2.6 for b) through the keyboard,

then the statement

int a, b, sum;

should be replaced by the statement

float a, b, sum;

and the statement

printf ("the sum of a and b = %d", sum); should be replaced by the statement printf ("the sum of a and b = %f", sum);

i.e.,

```
#include<stdio.h>
int main ()
{
float a, b, sum;
a=1.5;
b=2.6;
sum = a + b;
printf ("the sum of a and b = %f", sum);
return 0;
}
```

The output on the screen:

```
the sum of a and b = 4.1
```

The statement

printf ("the sum of a and b = %f", sum);

make provision to print the output:

the sum of a and b = 4.1

In the statement

printf ("the sum of a and b = %f", sum);

format string %f tells the printf function to print a floating point value which is sum.

Since a = 1.5 and b = 2.6 therefore:

the sum of a and b = 1.5 + 2.6 = 4.1 which is outputted on the screen.

If the statement

printf ("the sum of a and b = %f", sum);

is replaced by the statement

printf ("the sum of a and b = %f, sum");

Then output on the screen is:

the sum of a and b = %f, sum

And if the statement printf ("the sum of a and b = %f", sum); is omitted from the C program,

then the program will be successfully executed but there will be no display of the output on the screen.

If you want to supply the values for a and b through the key board, then the statements

a=1.5;
b=2.6;

should be replaced by the statements

printf ("Enter any two numbers:");
scanf ("%f %f", &a, &b);

i.e., the program should be rewritten as:

```
#include<stdio.h>
int main ()
{
float a, b, sum;
printf ("Enter any two numbers:");
scanf ("%f %f", &a, &b);
sum = a+ b;
printf ("the sum of a and b = %f", sum);
return 0;
}
```

The output on the screen:

Enter any two numbers:

If you enter two numbers 2.9 & 3.6

the sum of a and b = 6.5 will be outputted on the screen with trailing zeros to fill up memory.

As Said Earlier:

ampersand ("&") imply the address and

&a and &b imply the addresses of the created float variables a and b stored in the computer memory i.e., when we enter a number for a and b through the keyboard, these numbers are read by scanf () function and they are stored in the computer memory (i.e., the number entered for a is stored in the address of a (i.e., stored in &a) and the number entered for b is stored in the address of b (i.e., stored in &b)).

There are 2 format strings in the statement

scanf ("%f %f", &a, &b);

one format string %f corresponds to &a i.e., %f tells the scanf () function to read the number entered through the keyboard for a and store it in the address of a in the computer memory.

and the other format string %f corresponds to &b i.e., %f tells the scanf () function to read the number entered through the keyboard for b and store it in the address of b in the computer memory.

If the two format strings are separated by a comma i.e.,

scanf ("%f, %f", &a, &b);

Then the compilation error will be displayed on the screen.

Note:

The statement printf ("Enter any two numbers:"); make provision to print

Enter any two numbers:

on the screen and the statement scanf ("%f %f", &a, &b); read the two numbers 2.9 and 3.6 entered through the keyboard and store them in the computer memory.

If the statements

printf ("Enter any two numbers:");
scanf ("%f %f", &a, &b);

are replaced by the statements:

printf ("Enter any number:");
scanf ("%f", &a);
printf ("Enter any number:");
scanf ("%f", &b);

i.e.,

```
#include<stdio.h>
int main ()
{
float a, b, sum;
printf ("Enter any number:");
scanf ("%f", &a);
printf ("Enter any number:");
scanf ("%f", &b);
sum = a+ b;
printf ("the sum of a and b = %f", sum);
return 0;
}
```

Then the output on the screen:

Enter any number:

If you enter a number 2.9

Enter any number:

If you enter a number 3.6

the sum of a and b = 6.5 will be outputted on the screen.

If the statement

```
printf ("the sum of a and b = %f", sum);
```

is replaced by the statement

```
printf ("the sum of %f and %f = %f", a, b, sum);
```

Then the output on the screen is:

```
the sum of 2.9 and 3.6 = 6.5
```

In the statement

```
printf ("the sum of %f and %f = %f", a, b, sum);
```

there are three format strings:

The format string %f after the statement (the sum of) indicates that the value to be displayed needs to be taken from a variable a.

The format string %f after the statement (the sum of %f and) indicates that the value to be displayed needs to be taken from a variable b.

The format string %f after the statement (the sum of %f and %f =) indicates that the value to be displayed needs to be taken from a variable sum.

Program 1.4

C program to convert the temperature in Celsius to Fahrenheit

```
#include<stdio.h>
int main ()
{
float C, F;
C=38.5;
F = 9*C/5 +32;
printf ("temperature in Fahrenheit= %f", F);
return 0;
}
```

The output on the screen:

```
temperature in Fahrenheit= 101.3
```

Note: If x is used instead of * and F = 9C/5 +32 is used of F = 9*C/5 +32, then the compilation error will be displayed on the screen.

If you want to supply a number 16 digits after decimal point i.e., 36.5555555555555555 for C, then the statement

double C, F; should be used instead of the statement float C, F;

and %lf should be used instead of %f

i.e.,

```
#include<stdio.h>
int main ()
{
double C, F;
C=38.5555555555555555;
F = 9*C/5 +32;
printf ("temperature in Fahrenheit= %lf", F);
return 0;
}
```

And if you want to supply the number 16 digits after decimal point for C through the key board, then the statement

```
C = 38.5;
```

should be replaced by the statements:

```
printf ("Enter any number:");
scanf ("%lf",      &   C);
```

i.e.,

```
#include<stdio.h>
int main ()
{
double C, F;
printf ("Enter any number:");
scanf ("%lf",      &   C);
F = 9*C/5 +32;
printf ("temperature in Fahrenheit= %lf", F);
return 0;
}
```

Note:

```
#include <stdio.h>
int main ()
{
double C, F;
C = 25.3333333333333333;
F = 9*C/5 +32;
printf ("temperature in Fahrenheit= %lf", F);
}
```

The output on the screen:

```
temperature in Fahrenheit = 77.600000
```

If the statement double C, F; is replaced by the statements

```
double C;
float F;
```

i.e., if the above program is rewritten as:

```
#include <stdio.h>
int main ()
{
double C;
float F;
C = 25.3333333333333333;
F = 9*C/5 +32;
printf ("temperature in Fahrenheit= %f", F); // %f is used because the data type for F is float
return 0;
}
```

Then there is slight change in the output on the screen:

```
temperature in Fahrenheit = 77.599998
```

Write a program to print the sum of three numbers

Answer:

```
#include<stdio.h>
int main ()
{
int a, b, c, sum;
printf ("Enter any three numbers:");
scanf ("%d %d%d", & a, & b, & c);
sum = a + b + c;
printf ("the sum of a, b and c = %d", sum);
return 0;
}
```

Write a program to print the Equivalent hexadecimal value of an integer

Answer:

```
#include<stdio.h>
int main ()
{
int a = 45;
printf ("%x", a);
return 0;
}
```

Output on the screen:

```
2d
```

Write a program to print the area of a triangle, given

area = (s (s-a) (s-b) (s-c))1/2 where s = (a + b + c) / 2

Answer:

```
#include<stdio.h>
#include<math.h>
int main ()
{
int a, b, c, s, area;
a = 3;
b= 4;
c=5;
s = (a + b + c) / 2;
area = sqrt ((s * (s-a) * (s-b) * (s-c));
printf ("the area of the triangle = %d", area);
return 0;
}
```

Note: since sqrt () is not part of C language or of standard input output file i.e., (stdio.h file), it is part of math file i.e., (math.h file which defines various mathematical functions) the statement #include<math.h> should be included in the C program otherwise the compilation error will be flagged on the screen stating that sqrt () is not declared or defined.

Note: If the statement area = (s (s-a) (s-b) (s-c)) 1/2 is written instead of

area = sqrt ((s * (s-a) * (s-b) * (s-c));

Then the compilation error will be displayed on the screen because C does not support

area = (s (s-a) (s-b) (s-c)) 1/2.

Stuff you need to know about:

1 kilobyte = 1024 bytes
1 megabyte = 1024 1024 bytes
1 gigabyte = 1024 1024 1024 bytes

Program 1.5

C program to find the product of two numbers

```
#include<stdio.h>
int main ()
{
int a, b, product;
a=1;
b=2;
product = a * b;
printf ("the product of a and b = %d", product);
return 0;
}
```

The output on the screen:

```
the product of a and b = 2
```

If you want to insert a 10 digit number for a and b i.e.,

a=1000000000

b=3000000000, then the statement:

int a, b, product; should be replaced by the statement long int a, b, product;

and %ld should be used instead of %d

i.e., the program should be rewritten as:

```
#include<stdio.h>
int main ()
{
long int a, b, product;
a=1000000000;
b=2000000000;
product = a * b;
printf ("the product of a and b = %ld", product);
return 0;
}
```

The output on the screen:

```
the product of a and b = 3000000000000000000
```

If you want to supply the values for a and b through the key board, then the statements

a=1;

b=2; should be replaced by the statements

```
printf ("Enter any two numbers:");
scanf ("%d %d", & a, & b);
```

i.e.,

```
#include<stdio.h>
int main ()
{
int a, b, product;
printf ("Enter any two numbers:");
scanf ("%d%d", & a, & b);
product = a* b;
printf ("the product of a and b = %d", product);
return 0;
}
```

The output on the screen:

```
Enter any two numbers:
```

If you enter two numbers 1 and 3

the product of a and b = 3 will be outputted on the screen.

If you replace the statements

```
printf ("Enter any two numbers:");
scanf ("%d%d",     & a,     & b);
```

by the statements

```
printf ("Enter any number:");
scanf ("%d",     & a);
printf ("Enter any number:");
scanf ("%d",     & b);
```

Then the output on the screen will be:

```
Enter any number:
```

If you enter the number 3

```
Enter any number:
```

If you enter the number 3

the product of a and b = 9

will be outputted on the screen.

If the statement

```
printf ("the product of a and b = %d"; product);
```

is written instead of the statement

```
printf ("the product of a and b = %d", product);
```

i.e., instead of variable separator (i.e., comma) semicolon is used -- Then the compilation error will be displayed on the screen.

Note:

```
#include <stdio.h>
int main ()
{
printf ("Hello, World!");
printf ("Hello, World!\b");
printf ("Hello, World!\b");
printf ("Hello, World!\b");
return 0;
}
```

i.e., if back space \b is used then

Hello, World!Hello, World!Hello, World!Hello, World!

will be outputted on the screen.

If carriage return \r is used instead of \b

i.e.,

```
#include <stdio.h>
int main ()
{
printf ("Hello, World!");
printf ("Hello, World!\r");
printf ("Hello, World!\r");
printf ("Hello, World!\r");
return 0;
}
```

The output on the screen is:

```
Hello, World!Hello, World!
Hello, World!
Hello, World!
```

If Horizontal tab \t is used instead of \r

i.e.,

```
#include <stdio.h>
int main ()
{
printf ("Hello, World!\t");
printf ("Hello, World!\t");
printf ("Hello, World!\t");
printf ("Hello, World!\t");
return 0;
}
```

The output on the screen is:

```
Hello, World!  Hello, World!  Hello,       World!
Hello, World!
```

If vertical tab \v is used instead of \t

i.e.,

```
#include <stdio.h>
int main ()
{
printf ("Hello, World!\v");
printf ("Hello, World!\v");
printf ("Hello, World!\v");
printf ("Hello, World!\v");
return 0;
}
```

The output on the screen is:

```
Hello, World!
Hello, World!
Hello, World!
Hello, World!
```

Program 1.5

C program to find the square of a number

```
#include<stdio.h>
int main ()
{
int a, b;
a=2;
b = a * a;
printf ("the square of a = %d", b);
}
```

The output on the screen:

```
the square of a = 4
```

If the statement b = a * a; is replaced by b = pow ((a), 2);

i.e., if the above program is rewritten as:

```
#include<stdio.h>
#include<math.h>
int main ()
{
int a, b;
a=2;
b = pow ((a), 2);
printf ("the square of a = %d", b);
return 0;
}
```

Then there will be no display of compilation error on the screen or there will be no change in the output on the screen i.e., the square of a = 4 will be outputted on the screen.

which means:

b = pow ((a), 2); is the same as b = a*a;

Since b = pow ((a), 2); is used instead of b = a*a;

#include<math.h> should be included in the C program as b = pow ((a), 2); is supported by #include<math.h>

Otherwise compilation Error will be displayed on the screen.

If you want to supply the integer value for a through the key board, then the statement

a=2; is replaced by the statements

```
printf ("Enter any number:");
scanf ("%d",   &  a);
```

i.e.,

```
#include<stdio.h>
int main ()
{
int a, b;
printf ("Enter any number:");
scanf ("%d",   &  a);
b = a * a;
printf ("the square of a = %d", b);
return 0;
}
```

The output on the screen:

Enter any number:

If you enter a number 4

the square of a = 16 will be outputted on the screen.

Note:

If scanf (%d,   & a); is written instead of scanf ("%d",   & a);

If printf (the square of a = %d, b); is written instead of printf ("the square of a = %d", b);

If the main function is followed by a semicolon i.e.,

int main (); is written instead of int main ()

Then the compilation error will be displayed on the screen.

But if the body of the main function is followed by a semicolon i.e.,

```
int main ()
{
}; is written instead of
int main ()
{
}
```

There will be no display of the compilation error on the screen.

```
int main (); ERROR
int main ()
{
}; NO ERROR
```

Write a program to print the cube of a number

Answer:

```
#include<stdio.h>
#include<math.h>
int main ()
{
int a, b;
a=2;
b = pow ((a), 3);
printf ("the cube of a = %d", b);
return 0;
}
```

Write a program to print the energy of the substance using energy = mc2

Answer:

```
#include<stdio.h>
#include<math.h>
int main ()
{
int m;
long int c, energy;
m=2;
c = 300000000;
energy = m * pow ((c), 2);
printf ("the energy of the substance = %ld joules", energy);
return 0;
}
```

Program 1.6

C program to find the greatest of two numbers using if - else statement

The syntax of if else statement (Conditional Statements):

```
if (this condition is true)
{
print this statement;
}
else
{
print this statement;
}
```

```
#include<stdio.h>
int main ()
{
int a, b;
a = 2;
b = 3;
if (a>b)
{
printf ("a is greater than b");
}
else
{
printf ("b is greater than a");
}
return 0;
}
```

The output on the screen:

```
b is greater than a
```

Since the condition a>b within the parentheses is not true, the statement a is greater than b is not executed;

instead the execution skips and pass to print the statement b is greater than a.

In simpler words,

(a>b) is the condition (i.e., logical expression that results in true or false) and

if the condition (a>b) is true, then the statement:

```
{
printf ("a is greater than b");
}
```

is executed to print the output:

```
a is greater than b
```

else the statement

```
{
printf ("b is greater than a");
}
```

is executed to print the output:

```
b is greater than a
```

If you want to supply the integer values for a and b through the key board, then the statements

```
a=2;
b=3;
```

should be replaced by the statements

```
printf ("Enter any number:");
scanf ("%d",    &  a);
printf ("Enter any number:");
scanf ("%d",    &  b);
```

i.e., the program should be rewritten as:

```
#include<stdio.h>
int main ()
{
int a, b;
printf ("Enter any number:");
scanf ("%d",    &  a);
printf ("Enter any number:");
scanf ("%d",    &  b);
if (a>b)
{
printf ("a is greater than b");
}
else
{
printf ("b is greater than a");
}
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter the number 6
Enter any number:
If you enter the number 3
a is greater than b
```

will be outputted on the screen.

Program 1.7

C program to find the greatest of three numbers using if - else if - else statement

The syntax of if - else if - else statement:

```
if (this condition is true)
{
print this statement;
}
else if (this condition is true)
{
print this statement;
}
else
{
print this statement;
}
```

```
#include<stdio.h>
int main ()
{
int a, b, c;
printf ("Enter any number:");
scanf ("%d",    &  a);
printf ("Enter any number:");
scanf ("%d",    &  b);
printf ("Enter any number:");
scanf ("%d",    &  c);
if (a>b     &    &  a>c)
{
printf ("%d is greater than %d and %d", a, b, c);
}
else if (b>a     &    &  b>c)
{
printf ("%d is greater than %d and %d", b, a, c);
}
else
{
printf ("%d is greater than %d and %d", c, b, a);
}
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter the number 2
Enter any number:
If you enter the number 3
Enter any number:
If you enter the number 4
4 is greater than 3 and 2
```

will be outputted on the screen.

double ampersand " &   & " imply and.

(a>b &   & a>c)

(b>a &   & b>c)

denote conditions.

i.e., the condition

(a>b &   & a>c) imply:

a is greater than b and a is greater than c

and if this condition is true, then the statement

```
{
printf ("a is greater than b and c");
}
```

is executed to print the output:

```
a is greater than b and c
```

and if the condition (a>b & & a>c) is not true

the statement

```
{
printf ("a is greater than b and c");
}
```

is not executed; instead the execution skips and pass to the condition (b>a & & b>c)

and if this condition is true, then the statement

```
{
printf ("b is greater than a and c");
}
```

is executed to print the output:

```
b is greater than a and c
```

and if the condition (b>a & & b>c) is not true, then the statement

```
{
printf ("b is greater than a and c");
}
```

is not executed; instead the execution skips and the statement

```
{
printf ("c is greater than b and a");
}
```

is executed to print the output:

```
c is greater than b and a
```

If the statements:

```
if (a>b & & a>c)
{
printf ("%d is greater than %d and %d", a, b, c);
}
else if (b>a & & b>c)
{
printf ("%d is greater than %d and %d", b, a, c);
}
else
{
printf ("%d is greater than %d and %d", c, b, a);
}
```

are replaced by the statements:

```
if (a>b & & a>c)
printf ("%d is greater than %d and %d", a, b, c);
else if (b>a & & b>c)
printf ("%d is greater than %d and %d", b, a, c);
else
printf ("%d is greater than %d and %d", c, b, a);
```

i.e., if the program is rewritten as:

```
#include<stdio.h>
int main ()
{
int a, b, c;
printf ("Enter any number:");
scanf ("%d", & a);
printf ("Enter any number:");
scanf ("%d", & b);
printf ("Enter any number:");
scanf ("%d", & c);
if (a>b & & a>c)
printf ("%d is greater than %d and %d", a, b, c);
else if (b>a & & b>c)
printf ("%d is greater than %d and %d", b, a, c);
else
printf ("%d is greater than %d and %d", c, b, a);
return 0;
}
```

There will be no display of error on the screen

c is greater than b and a

will be successfully outputted on the screen

What will be the output of the following program?

```
#include <stdio.h>
int main ()
{
int a, b;
a=2;
b=2;
if (a>b || a= = b)
printf ("a is greater than or equal to b");
else
printf ("b is greater than a");
return 0;
}
```

Answer:

```
a is greater than or equal to b
```

Note: symbol || denote OR i.e., a>b || a= = b denote a is greater than or a is equal to b.

Program 1.8

C program to find the average of 10 numbers

```
#include<stdio.h>
int main ()
{
int N1, N2, N3, N4, N5, N6, N7, N8, N9, N10, X;
printf ("Enter any 10 numbers:");
scanf ("%d%d%d%d%d%d%d%d%d%d", & N1, & N2, & N3, & N4, & N5, & N6, & N7, & N8, & N9, & N10);
X = (N1 + N2 + N3 + N4 + N5 + N6 + N7 + N8 + N9 + N10) /10;
printf ("the average of 10 numbers = %d", X);
return 0;
}
```

The output on the screen:

```
Enter any 10 numbers:
```

If you enter ten numbers 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10

the average of 10 numbers = 5

will be outputted on the screen.

Note: The average of 10 numbers is 5.5, the output on the screen is 5 because the data type int is used instead of float.

Any mathematical expression should be written in C equivalent expression to prevent the display of compilation error on the screen because C language does not accept the general mathematical expressions.

Mathematical expression	C equivalent expression
x y / z	x * y / z
(ax + 1) (by + 2)	(a * x + 1) * (b * y + 2)
(a+b)2/ (a-b)2	(a+b) * (a+b) / (a-b) * (a-b) or pow ((a+b), 2) / pow ((a - b), 2)
log10(x/y + c)	log 10 (x/y + c)
ax2+bx+c	a*x*x+b*x+c
lnx	log (x)
ex + b	exp (x) + b
sin + cos	sin (theta) + cos (theta)
= +	alpha = beta + gamma
x1/2	sqrt (x)
x1/3	cbrt (x)
(p2+ q2)1/2	sqrt (p*p + q*q)
2a2+ 3b	2a *a + 3b + 2
a = e x / (1+ sin)1/2	a = exp (x / sqrt (1 + sin (theta)))

What will be the output of the following programs:

(a)

```
#include <stdio.h>
#include<math.h>
int main ()
{
int a, b, x;
x=2;
b=2;
a = exp (x) + b;
printf ("the value of a = %d", a);
return 0;
}
```

Answer:

the value of a = 9

(b)

```
#include <stdio.h>
#include<math.h>
int main ()
{
int alpha, beta, gamma;
alpha =2;
beta=2;
gamma= 2 * alpha + beta;
printf ("the value of alpha = %d", alpha);
return 0;
}
```

Answer:

the value of alpha = 2

(c)

```
#include <stdio.h>
#include<math.h>
int main ()
{
double theta, result;
theta = 90;
result = sin (theta);
printf ("The sine 90 degrees is = %lf ", result);
return 0;
}
```

Answer:

The sine 90 degrees is = 0.893997

What is C equivalent expression of (x/y) n-1?

Answer: pow ((x/y), n-1)

Program 1.9

C program to find the square root of a number

```
#include<stdio.h>
#include<math.h>
int main ()
{
int a, b;
printf ("Enter any number:");
scanf ("%d",    &   a);
b = sqrt (a);
printf ("the square root of a number = %d", b);
return 0;
}
```

The output on the screen:

Enter any number:

If you enter the number 4

the square root of a number = 2

is outputted on the screen.

Suppose if you enter the number 2, the square root of a number = 1 is outputted on the screen because int is used instead of float.

Note:

Since b = sqrt (a) is written

#include<math.h>

must be included in the above program otherwise compilation error will flag on the screen.

i.e., the program:

```
#include<stdio.h>
int main ()
{
int a, b;
printf ("Enter any number:");
scanf ("%d",    &   a);
b = sqrt (a);
printf ("the square root of a number = %d", b);
return 0;
}
```

will flag compilation error on the screen.

If float is used instead of int then the above program take the form:

```
#include<stdio.h>
#include<math.h>
int main ()
{
float a, b;
printf ("Enter any number:");
scanf ("%d",    &   a);
b = sqrt (a);
printf ("the square root of a number = %f", b);
return 0;
}
```

The output on the screen:

```
Enter any number:
```

If you enter the number 5

```
the square root of a number = 2.23
```

is outputted on the screen.

This program can also be written as:

```
#include<stdio.h>
#include<math.h>
int main ()
{
printf ("the square root of a number = %f", sqrt (4));
return 0;
}
```

//--

|| imply or
> imply greater than
<imply less than
= = imply equal to
! imply not
!= imply not equal to
&   & imply and
& imply address

--//

Program 2.0

C program to find the simple interest

```
#include<stdio.h>
int main ()
{
int P,T, R, SI;
P = 1000;
T = 2;
R = 3;
SI = P*T*R/100;
printf ("the simple interest = %d", SI);
return 0;
}
```

The output on the screen:

```
the simple interest = 60
```

Note:

If you write SI = PTR/100; instead of SI = P*T*R/100;

Then compilation error is displayed on the screen because C language does not accept the general expressions.

If you want to supply the values for P, T and R through the key board, then the statements:

```
P = 1000;
T = 2;
R = 3;
```

should be replaced by the statements:

```
printf ("Enter any number:");
scanf ("%d",    &   P);
printf ("Enter any number:");
scanf ("%d",    &   T);
printf ("Enter any number:");
scanf ("%d",    &   R);
```

i.e., the above program should take the form:

```
#include<stdio.h>
int main ()
{
int P,T, R, SI;
printf ("Enter principal amount:");
scanf ("%d",    &   P);
printf ("Enter time:");
scanf ("%d",    &   T);
printf ("Enter rate of interest:");
scanf ("%d",    &   R);
SI = P*T*R/100;
printf ("the simple interest = %d", SI);
return 0;
}
```

The output on the screen:

```
Enter principal amount:
```

If you enter the principal amount 1000

```
Enter time:
```

If you enter the time 2

```
Enter rate of interest:
```

If you enter the rate of interest 3

```
the simple interest = 60
```

will be outputted on the screen.

Note: If write the statement scanf ("%d," & P); instead of scanf ("%d",   &   P);

or

if write the statement scanf (%d,   &   P); instead of scanf ("%d",   &   P); i.e., format string for data type int i.e., %d is not enclosed by double quotes ("")

Then compilation error will be displayed on the console screen.

Program 2.1

C program to find whether the person is senior citizen or not

```
#include<stdio.h>
int main ()
{
int age;
age=20;
if (age> = 60)
{
printf ("senior citizen");
}
if (age<60)
{
printf ("not a senior citizen");
}
return 0;
}
```

The output on the screen:

```
not a senior citizen
```

(age> = 60) means age greater than or equal to 60

If you want to supply the value for age through the key board, then the statement

```
age=20;
```

should be replaced by the statements:

```
printf ("Enter age:");
scanf ("%d",     &   age);
```

i.e., the above program should take the form:

```
#include<stdio.h>
int main ()
{
int age;
printf ("Enter age:");
scanf ("%d",     &   age);
if (age>60)
{
printf ("senior citizen");
}
if (age<60)
{
printf ("not a senior citizen");
}
return 0;
}
```

The output on the screen:

```
Enter age:
```

If you enter the value 60

senior citizen will be outputted on the screen.

Suppose if you enter the value 27

not a senior citizen will be outputted on the screen.

Program 2.2

C program to get marks for 3 subjects and declare the result.

If the marks >= 35 in all the subjects the student passes else fails.

```
#include<stdio.h>
int main ()
{
int M1, M2,M3;
M1 = 38;
M2= 45;
M3 = 67;
if (M1>= 35     &   &   M2>= 35        &
&     M3>= 35)
{
printf ("candidate is passed");
}
else
{
printf ("candidate is failed");
}
return 0;
}
```

The output on the screen:

```
candidate is passed
```

>= imply greater than or equal to and double ampersand imply and

(M1>= 35   &   &   M2>= 35   &
&   M3>= 35) denote the condition and this condition imply M1 is greater than or equal to 35

and M2 is greater than or equal to 35 and M3 is greater than or equal to 35. And if this condition is TRUE, then the statement

```
{
printf ("candidate is passed");
}
```

is executed to print the output:

```
candidate is passed
```

else the statement

```
{
printf ("candidate is failed");
}
```

is executed to print the output:

```
candidate is failed
```

If you want to supply the integer values for marks M1, M2 and M3 through the key board, then the statements:

```
M1 = 38;
M2= 45;
M3 = 67;
```

should be replaced by the statements:

```
printf ("Enter any three numbers:");
scanf ("%d%d%d",   &  M1,      &  M2,
 &  M3);
```

i.e.,

```
#include<stdio.h>
int main ()
{
int M1, M2,M3;
printf ("Enter any three numbers:");
scanf ("%d%d%d",   &  M1,      &  M2,
 &  M3);
if (M1>= 35 & & M2>= 35 &
& M3>= 35)
{
printf ("candidate is passed");
}
else
{
printf ("candidate is failed");
}
return 0;
}
```

The output on the screen:

```
Enter any three numbers:
```

If you enter three numbers 26, 28, 39

candidate is failed will be outputted on the screen.

Header file in C	the functions it defines
stdio.h	(standard input output header file) standard input output functions (like scanf and printf functions)
math.h	mathematical functions (like log (), sqrt (), sin (), cos (), log10() etc.)
stdlib.h	standard library functions (like void abort (void) - a function which causes an abnormal/ unusual program termination)
ctype.h	character manipulation functions (like isalpha () which checks whether a character is an alphabet or not)
graphics.h	graphical functions
conio.h	(console input output header file) console input output functions like clrscr () - a function which clears the screen.

Note: The term console usually refers to monitor or display screen.

Write a program to check whether a character is an alphabet or not using the function isalpha ()

```
#include <stdio.h>
#include <ctype.h>
int main ()
{
int a =2;
if ( isalpha (a) )
{
printf (" the character a is an alphabet");
}
else
{
printf ("the character a is not an alphabet");
}
return 0;
}
```

The output on the screen:

```
the character a is not an alphabet
```

```
#include <stdio.h>
#include <ctype.h>
int main ()
{
char a = 'b';
if ( isalpha (a) )
{
printf (" the character a is an alphabet");
}
else
{
printf ("the character a is not an alphabet");
}
return 0;
}
```

The output on the screen:

```
the character a is an alphabet
```

If the statement char a = b; is written instead of char a = 'b'; Then the compilation error will be flagged on the display screen.

Program 2.3

C program to find profit or loss

```
#include<stdio.h>
int main ()
{
int CP, SP, loss, profit;
printf ("Enter cost price:");
scanf ("%d", & CP);
printf ("Enter selling price:");
scanf ("%d", & SP);
if (SP>CP)
{
printf ("profit=%d", SP-CP);
}
else
{
printf ("loss =%d", CP-SP);
}
return 0;
}
```

The output on the screen:

```
Enter cost price:
```

If you enter the cost price 25

```
Enter selling price:
```

If you enter the selling price 26

profit = 1 will be outputted on the screen.

If the condition (SP>CP) is true, then the statement

```
{
printf ("profit=%d", SP-CP);
}
```

is executed to print the output:

profit = SP-CP (in this case profit = 26-25 =1)

else the statement

```
{
```

```
printf ("loss=%d", CP-SP);
}
```

is executed to print the output:

```
loss = CP-SP
```

Program 2.4

C program to convert inches into centimeter

```
#include<stdio.h>
int main ()
{
float I, C;
I=3.5;
C = 2.54*I;
printf ("length in centimeters= %f", C);
return 0;
}
```

The output on the screen:

```
length in centimeters = 8.89
```

Note: float is used instead of int because I = 3.5 if int is used instead of float then the result will not be clearly outputted i.e., instead of 8.89 the computer displays only 8. And since float is used instead of int, the operator %d is replaced by the operator %f.

If you want to supply the floating value for I through the key board, then the above program should take the form:

```
#include<stdio.h>
int main ()
{
float I, C;
printf ("Enter the length in inches:");
scanf ("%f",      &   I);
C = 2.54*I;
printf ("length in centimeters= %f", C);
return 0;
}
```

The output on the screen:

```
Enter the length in inches:
```

If you enter the floating point value or fractional or decimal number for I i.e., 25.5

length in centimeters = 64.9 will be outputted on the screen.

Program 2.5

C program to find the incremented and decremented values of two numbers.

```
#include<stdio.h>
int main ()
{
int a, b, c, d, e, f;
a = 10;
b=12;
c=a+1;
d=b+1;
e=a-1;
f=b-1;
printf ("the incremented value of a =%d", c);
printf ("the incremented value of b =%d", d);
printf ("the decremented value of a =%d", e);
printf ("the decremented value of b =%d", f);
return 0;
}
```

The output on the screen:

```
the incremented value of a = 11 the incremented value of b = 13 the decremented value of a = 9 the decremented value of b = 11
```

If the statements:

```
printf ("the incremented value of a =%d", c);
printf ("the incremented value of b =%d", d);
printf ("the decremented value of a =%d", e);
printf ("the decremented value of b =%d", f);
```

are replaced by the statements:

```
printf ("the incremented value of a =%d\n", c);
printf ("the incremented value of b =%d\n", d);
printf ("the decremented value of a =%d\n", e);
printf ("the decremented value of b =%d\n", f);
```

Then the output on the screen is:

```
the incremented value of a = 11
the incremented value of b = 13
the decremented value of a = 9
the decremented value of b = 11
```

Note:

Even if the statements:

```
printf ("the incremented value of a =%d\n", c);
printf ("the incremented value of b =%d\n", d);
printf ("the decremented value of a =%d\n", e);
printf ("the decremented value of b =%d\n", f);
```

are replaced by the statements:

```
printf ("\n the incremented value of a =%d", c);
printf ("\n the incremented value of b =%d", d);
printf ("\n the decremented value of a =%d", e);
printf ("\n the decremented value of b =%d", f);
```

There will be no change in the output on the screen.

If you want to supply the values for a and b through the key board, then the above program should take the form:

```
#include<stdio.h>
int main ()
{
int a, b, c, d, e, f;
printf ("Enter any number:");
scanf ("%d",      &   a);
printf ("Enter any number:");
scanf ("%d",      &   b);
c=a+1;
d=b+1;
e=a-1;
f=b-1;
printf ("the incremented value of a =%d\n", c);
printf ("the incremented value of b =%d\n", d);
printf ("the decremented value of a =%d\n", e);
printf ("the decremented value of b =%d\n", f);
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter the number 2
Enter any number:
If you enter the number 3
the incremented value of a = 3
the incremented value of b = 4
the decremented value of a = 1
the decremented value of b = 2
will be outputted on the screen.
```

Note: b++ is same as b + 1 and b-- is same as b - 1.

Program 2.6

The percentage marks are entered and the grades are allotted as follows:

percentage>= 60 First Class

percentage>=50 and per <= 60 Second Class

percentage>= 40 and per <= 50 Pass Class

percentage< 40 Fail

Write a C program for the above:

```
#include<stdio.h>
int main ()
{
int P;
printf ("Enter the percentage:");
scanf ("%d",   &  P);
if (P >= 60)
{
printf ("first class");
}
if (P>=50  &  & P <60)
{
printf ("second class");
}
else if (P>=40  &  & P<=50 )
{
printf ("pass class");
}
else
{
printf ("fail");
}
return 0;
}
```

The output on the screen:

```
Enter the percentage:
If you enter the percentage 65
first class will be outputted on the screen.
```

Program 2.7

C program to calculate the discounted price and the total price after discount

Given:

If purchase value is greater than 1000, 10% discount

If purchase value is greater than 5000, 20% discount

If purchase value is greater than 10000, 30% discount

discounted price

```
#include<stdio.h>
int main ()
{
double PV, dis;
printf ("Enter purchased value:");
scanf ("%lf",   &  PV);
if (PV>1000)
{
printf ("dis=%lf", PV* 0.1);
}
else if (PV>5000)
{
printf ("dis =%lf", PV* 0.2);
}
else
{
printf ("dis=%lf", PV* 0.3);
}
return 0;
}
```

The output on the screen:

```
Enter purchased value:
If you enter the purchased value 6500
dis = 1300.000000 will be outputted on the screen.
```

(PV>1000), (PV>5000) denote the conditions and if the condition (PV>1000) is true i.e., purchased value is greater than 1000, then the statement

```
{
printf ("dis=%d", PV* 0.1);
}
```

is executed to print the output:

dis= PV* 10% = PV* 10 /100 = PV* 0.1

and if the condition (PV>1000) is false and if the condition (PV>5000) is true i.e., purchased value is greater than 5000, then the statement

```
{
printf ("dis=%d", PV* 0.2);
}
```

is executed to print the output:

dis= PV* 20% = PV* 20 /100 = PV* 0.2

and if the condition (PV>5000) is not true i.e., purchased value is less than 5000, then the statement

```
{
printf ("dis=%d", PV* 0.3);
}
```

is executed to print the output:

dis= PV* 30% = PV* 30 /100 = PV* 0.3

total price

```
#include<stdio.h>
int main ()
```

```
{
double PV, total;
printf ("Enter purchased value:");
scanf ("%lf",     &   PV);
if (PV<1000)
{
printf ("total=%lf", PV - PV* 0.1);
}
else if (PV<5000)
{
printf ("total =%lf", PV- PV* 0.2);
}
else
{
printf ("total=%lf", PV- PV* 0.3);
}
return 0;
}
```

The output on the screen:

Enter purchased value:

If you enter the purchased value 650

total = 585.000000 will be outputted on the screen.

If the condition (PV>1000) is true i.e., purchased value is greater than 1000, then the statement

```
{
printf ("total = %d", PV - PV* 0.1);
}
```

is executed to print the output:

total =PV- dis = PV- PV*10% = PV- PV* 10 /100 = PV - PV * 0.1

and if the condition (PV>1000) is false and if the condition (PV>5000) is true i.e., purchased value is greater than 5000, then the statement

```
{
printf ("total = %d", PV - PV* 0.2);
}
```

is executed to print the output:

total =PV- dis = PV- PV*20% = PV- PV* 20 /100 = PV - PV * 0.2

and if the condition (PV> 5000) is not true i.e., purchased value is less than 5000, then the statement

```
{
printf ("total = %d", PV - PV* 0.3);
}
```

is executed to print the output:

total =PV- dis = PV- PV*30% = PV- PV* 30 /100 = PV - PV * 0.3

Now, Combing both the programs (above), we can write:

```
#include<stdio.h>
int main ()
{
double PV, dis, total;
printf ("Enter purchased value:");
scanf ("%lf",     &   PV);
if (PV>1000)
{
printf ("dis=%lf", PV* 0.1);
printf ("total=%lf", PV - PV* 0.1);
}
else if (PV>5000)
{
printf ("dis =%lf", PV* 0.2);
printf ("total=%lf", PV - PV* 0.1);
}
else
{
printf ("dis=%lf", PV* 0.3);
printf ("total=%lf", PV - PV* 0.1);
}
return 0;
}
```

The output on the screen:

Enter purchased value:

If you enter the purchased value 850

dis = 85.000000

total = 765.000000

will be outputted on the screen.

Program 2.8

C program to print the first ten natural numbers using for loop statement

```
#include<stdio.h>
int main ()
{
int i;
for (i=1; i<=10; i++)
printf ("value of i =%d", i);
return 0;
}
```

The output on the screen is:

value of i = 1 value of i = 2 value of i= 3 value of i= 4 value of i= 5 value of i= 6 value of i = 7 value of i= 8 value of i = 9 value of i = 10

for (i=1; i<=10; i++) denote the for loop statement and the syntax of the for loop statement is:

for (initialization; condition; increment)

Here:

i=1 denote initialization (i.e., from where to start)

i<=10 denote the condition (i.e., stop when 10 is reached)

i++ imply increment (which tells the value of i to increase by 1 each time the loop is executed) and i++ is the same as i+1.

When a for loop executes, the following occurs:

i = 1

Is the condition (i<=10) is true?

Yes because i=1

The statement printf ("value of i =%d", i); is executed to print the output:

value of i = 1
Now, the value of i is:
i = 1+1 = 2
Is the condition (i<=10) is true?
Yes because i=2
The statement printf ("value of i =%d", i); is executed to print the output:
value of i = 2
Now, the value of i is:
i = 2+1 = 3
Is the condition (i<=10) is true?
Yes because i=3
The statement printf ("value of i =%d", i); is executed to print the output:
value of i = 3
Now, the value of i is:
i = 3+1 = 4
Is the condition (i<=10) is true?
Yes because i=4
The statement printf ("value of i =%d", i); is executed to print the output:
value of i = 4
Now, the value of i is:
i = 4+1 = 5
Is the condition (i<=10) is true?
Yes because i=5
The statement printf ("value of i =%d", i); is executed to print the output:
value of i = 5
Now, the value of i is:
i = 5+1 = 6
Is the condition (i<=10) is true?
Yes because i=6
The statement printf ("value of i =%d", i); is executed to print the output:
value of i = 6
Now, the value of i is:
i = 6+1 = 7
Is the condition (i<=10) is true?
Yes because i=7
The statement printf ("value of i =%d", i); is executed to print the output:
value of i = 7
Now, the value of i is:
i = 7+1 = 8
Is the condition (i<=10) is true?
Yes because i=8
The statement printf ("value of i =%d", i); is executed to print the output:
value of i = 8
Now, the value of i is:
i = 8+1 = 9
Is the condition (i<=10) is true?
Yes because i=9
The statement printf ("value of i =%d", i); is executed to print the output:
value of i = 9
Now, the value of i is:
i = 9+1 = 10
Is the condition (i<=10) is true?
Yes because i=10
The statement printf ("value of i =%d", i); is executed to print the output:
value of i = 10
and stop because the condition i<=10 is achieved.
If the statement:
printf ("value of i =%d", i);
is replaced by the statement:
printf ("value of i =%d\n", i);
or
printf ("\n value of i =%d", i);

Then the output on the screen is:
value of i = 1
value of i = 2
value of i = 3
value of i = 4
value of i = 5
value of i = 6
value of i = 7
value of i = 8
value of i = 9
value of i = 10
If the
for loop statement:
for (i=2; i<=10; i++)
is written instead of the statement:
for (i=1; i<=10; i++), then the output on the screen is:
value of i = 2 value of i = 3 value of i= 4 value of i= 5 value of i= 6 value of i = 7 value of i= 8 value of i = 9 value of i= 10
If the for loop statement:
for (i=1; i<10; i++)
is written instead of the statement:
for (i=1; i<=10; i++), then the output on the screen is:
value of i = 1 value of i = 2 value of i= 3 value of i= 4 value of i= 5 value of i= 6 value of i = 7 value of i= 8 value of i = 9
(Note: the condition i<=10 tells to print till value of i =10 but the condition i<10 tells to print till value of i=9)
If the statement:
for (i=1; i=10; i++)
is written instead of the statement:
for (i=1; i<=10; i++), then the output on the screen is:
value of i = 10 value of i = 10 value of i = 10 value of i = 10 value of i= 10 value of i= 10 value of i = 10 value of i= 10 value of i = 10 value of i = 10 value of i = 10 value of i = 10 value of i = 10

```
value of i = 10  value of i = 10 (continues...).
```

Note:

If the statement:

printf ("value of i =%d", i); is replaced by the statement printf ("%d\n", i);

Then the output on the screen is:

```
1
2
3
4
5
6
7
8
9
10
```

What will be the output of the following program:

```
#include<stdio.h>
int main ()
{
int i;
for (i =1; i<=5; i ++)
printf ("Linux is not portable\n", i);
return 0;
}
```

Answer:

```
Linux is not portable
Linux is not portable
Linux is not portable
Linux is not portable
Linux is not portable
```

C program to print the first ten natural numbers using for while loop statement

The syntax of while loop statement is:

```
while (this is the condition)
{
execute this statement;
}
```

```
#include<stdio.h>
int main ()
{
int i = 1;
while (i<=10)
{
printf ("%d\n", i++);
}
return 0;
}
```

The output on the screen is:

```
1
2
3
4
5
6
7
8
9
10
```

(i<=10) is the condition and

The statement

printf ("%d\n", i++);

is repeatedly executed as long as a given condition (i<=10) is true.

If the statement:

int i=1;

is replaced by the statement:

int i;

Then the compilation error will be displayed on the console screen because initialization is not defined i.e., from where to start is not declared.

If the statement:

int i = 1;

is replaced by the int i = 0;

Then the output on the screen is:

```
0
1
2
3
4
5
6
7
8
9
10
```

Similarly if the statement int i = 0; is replaced by the int i = 7;

Then the output on the screen is:

```
7
8
9
10
```

C program to print first 10 numbers using do while loop statement

The syntax of do while loop statement is:

```
do
{
execute this statement;
}
while (this is the condition);
```

```
#include<stdio.h>
int main ()
{
int i =1;
do
{
printf (" i= %d\n", i++);
} while (i<=10);
return 0;
}
```

The output on the screen is:

```
i=1
i=2
i=3
i=4
i=5
i=6
i=7
i=8
i=9
i=10
```

The statement:

```
printf (" i= %d\n", i++);
```

is executed and then condition (i<=10) is checked. If condition (i<=10) is true then

The statement:

```
printf (" i= %d\n", i++);
```

is executed again. This process repeats until the given condition (i<=10) becomes false.

Why LOOP is USED?

If loop is not used then the C program to print first 10 numbers should be written as follows:

```
#include<stdio.h>
int main ()
{
printf ("\n i = 1");
printf ("\n i = 2");
printf ("\n i = 3");
printf ("\n i = 4");
printf ("\n i = 5");
printf ("\n i = 6");
printf ("\n i = 7");
printf ("\n i = 8");
printf ("\n i = 9");
printf ("\n i = 10");
return 0;
}
```

It takes pretty long time to write the code and the execution time is pretty long i.e., Because to reduce the time taken to write the code and to reduce the execution time -- loop statement is used.

Write a program to print:

Never test for an error condition you don't know how to handle

5 times using for loop statement.

Answer:

```
#include<stdio.h>
int main ()
{
int i;
for (i =1; i<=5; i ++)
printf ("Never test for an error condition you don't know how to handle \n");
return 0;
}
```

Note:

For the program:

```
#include<stdio.h>
int main ()
{
int i;
for (i=1; i=5; i++)
printf ("Linux is not portable");
return 0;
}
```

The output on the screen is:

```
Linux is not portable Linux is not portable Linux is not portable Linux is not portable Linux is not portable Linux is not portable Linux is not portable Linux is not portable Linux is not portable Linux is not portable Linux is not portable Linux is not portable Linux is not portable. continues
```

Program 2.9

C program to print the characters from A to Z using for loop, do while loop and while loop statement.

C program to print the characters from A to Z using for loop statement:

```
#include<stdio.h>
int main ()
{
char a;
for ( a='A'; a<='Z'; a++)
printf ("%c\n", a);
return 0;
}
```

The output on the screen:

```
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
W
X
Y
Z
```

char means the data type is character.

The statement

char a; imply that we are creating the character a.

Since char a is used. Therefore: the format specifier %c should be used instead of %d or %f otherwise error will be flagged on the screen.

If the statement for (a=A; a<=Z; a++) is written instead of the statement for (a='A'; a<='Z'; a++)

Then the compilation error will be displayed on the console screen.

C program to print the characters from A to Z using while loop statement:

```
#include<stdio.h>
int main ()
{
char a = 'A';
while (a<='Z')
{
printf ("%c\n", a++);
}
return 0;
}
```

C program to print the characters from A to Z using do while loop statement:

```
#include<stdio.h>
int main ()
{
char a = 'A';
do
{
printf (" %c\n", a++);
} while (a<='Z');
return 0;
}
```

Program 3.0

C program to print the given number is even or odd.

```
#include<stdio.h>
int main ()
{
int a;
printf ("Enter any number:");
scanf ("%d",    &   a);
if (a%2 = = 0)
{
printf ("the number is even");
}
else
{
printf ("the number is odd");
}
return 0;
}
```

The output on the screen:

Enter any number:

If you enter the number 4

the number is even will be outputted on the screen.

Mathematical symbol % denote modulus and (a%2 = = 0) is the condition and this condition imply: a divided by 2 yields reminder = 0.

For example: if you enter the number 4

Then a = 4

Then 4 divided by 2 yields the remainder = 0

Then the statement

```
{
printf ("the number is even");
}
```

is executed to print the output:

the number is even

(Note: in C language = = implies equal to)

Suppose if you enter the number 3

Then a = 3

Then 3 divided by 2 yields the remainder = 1

Then the statement

```
{
printf ("the number is odd");
}
```

is executed to print the output:

the number is odd

Data type	Storage size
char	1 byte
short int	2 byte
float, long int	4 byte
double, long double	8 byte

Program 3.1

C program to print the remainder of two numbers

```
#include<stdio.h>
int main ()
{
int a, b, c;
printf ("Enter any number:");
scanf ("%d",    &   a);
printf ("Enter any number:");
scanf ("%d",    &   b);
c = a%b;
printf ("the remainder of a and b = %d", c);
return 0;
}
```

The output on the screen:

Enter any number:

If you enter the number 3

Enter any number:

If you enter the number 2

the remainder of a and b = 1 will be outputted on the screen.

Since (a=3 and b=2). Therefore:

3 divided by 2 (i.e., a divided by b) yields the remainder equal to 1

If the statement:

printf ("the remainder of a and b = %d", c);

is replaced by the statement:

printf ("the remainder of %d and %d = %d", a, b, c);

Then the output on the screen is:

Enter any number:

If you enter the number 3

Enter any number:

If you enter the number 2

the remainder of 3 and 2 = 1 will be outputted on the screen.

Program 3.2

C program to check the equivalence of two numbers.

```
#include<stdio.h>
int main ()
{
int x, y;
printf ("Enter any number:");
scanf ("%d",      &   x);
printf ("Enter any number:");
scanf ("%d",      &   y);
if (x-y==0)
{
printf ("the two numbers are equivalent");
}
else
{
printf ("the number are not equivalent");
}
return 0;
}
```

The output on the screen:

Enter any number:

If you enter the number 2

Enter any number:

If you enter the number 2

the two numbers are equivalent will be outputted on the screen.

Since 2-2 is equal to 0 (i.e., x-y = = 0).

Therefore: the statement

```
{
printf ("the two numbers are equivalent");
}
```

is executed to print the output:

two numbers are equivalent

If you enter the integers 3 and 2

The output on the screen:

the two numbers are not equivalent

Since 3-2 is not equal to 0 (i.e., x-y!= 0). Therefore: the statement

```
{
printf ("the two numbers are not equivalent");
}
```

is executed to print the output:

two numbers are not equivalent

(as said earlier: in C language the symbol!= implies not equal to)

What is the mistake in the following program:

```
#include<stdio.h>
int main ()
{
int year;
year =1996;
if (year%4==0)
printf ("leap year");
else
printf ("not a leap year");
return 0;
}
```

Answer:

There is no mistake in the above program. The output on the screen is:

leap year

Since year=1996. Therefore:

1996 divided by 4 (i.e., year divided by 4) yields the remainder equal to 0.

The statement

printf ("leap year");

is executed to print the output:

leap year

If the year is = 1995. Then

1995 divided by 4 (i.e., year divided by 4) yields the remainder not equal to 0.

The statement

printf ("not a leap year");

is executed to print the output:

not a leap year

Note: for a year to be leap year, year divided by 4 should yield remainder = zero.

//

An algorithm must be seen to be believed.

Donald Knuth

You might have an algorithm for getting from office home, for making a chunk of code that calculates the terms of the Fibonacci sequence, or for finding what you're looking for in a retail store. Algorithms are the building blocks of computer programs or a sequence of a sequence of unambiguous instructions (the term 'unambiguous ' indicates that there is no room for subjective interpretation) that tells how the problem can be addressed and solved - - which is definitely overblown in their importance like road maps for accomplishing a given, well-defined automated reasoning task - - which always have a clear stopping point. Long division and column addition are examples that everyone is familiar with- even a simple function for adding two numbers is an implementation of a particular algorithm. Online grammar checking uses algorithms. Financial computations use algorithms. A search engine like Google uses search engine algorithms (for example, takes search strings of keywords as input, searches its associated database for relevant web pages, and returns results). In fact, it is difficult to think of a task performed by your computer that does not use computer procedures that are a lot like a recipes (called algorithms).

The algorithm to add two numbers entered by userwould look something like this:

Step 1: Start

Step 2: Declare variables num1, num2 and sum.

Step 3: Read values num1 and num2.

Step 4: Add num1 and num2 and assign the result to sum.

sumnum1+num2

Step 5: Display sum

Step 6: Stop

Algorithms are the heart of computer science (usually means a procedure or basically an instance of logic written in software that solves a recurrent problem of finding an item with specific properties among a collection of items or transforming data according to specified actions to protect it), and the subject has countless practical applications as well as intellectual depth that is widely used throughout all areas of information technology including solving a mathematical problem (as of finding the greatest common divisor) in a finite number of steps that frequently involves repetition of an operation. The word algorithm - a mathematical concept whose roots date back to 600 AD with the invention of the decimal system -- derives from the name of the Muslim mathematician and astronomer, Mohammed ibn-Musa al-Khwarizmi, who was part with the royal court in Baghdad and who lived from about 780 to 850. The use of computers, however, has elevated the use of algorithms in daily transactions (like accessing an automated teller machine (ATM), booking an air or train or buying something online) to unprecedented levels of real-world problems with solutions requiring advanced algorithms abounds. And their use is only likely to grow. Many of the problems, though they may not seem realistic, require the set of well-defined algorithmic knowledge that comes up every day in the real world. By developing a good understanding of a series of logical steps in an algorithmic language, you will be able to choose the right one for a problem and apply it properly.

The algorithm is written in human readable and understandable form. To search an element in a given array, it can be done in two ways: Linear search and Binary search.

Linear Search:

Linear search is a very basic and simple search algorithm. In this type of search, a sequential search is made over all elements one by one. Every element is checked and if a match is found then that particular element is returned, otherwise the search continues till the end of the data collection.

For Example:

To search the element 17 it will go step by step in a sequence order:

8	10	12	15	17	20	25

=

17

Match not found

8	10	12	15	17	20	25

=

17

Match not found

8	10	12	15	17	20	25

=

17

Match not found

8	10	12	15	17	20	25

=

17

Match not found

8	10	12	15	17	20	25

=

17

Match found

Element 17 is returned

Linear search (whose running time increases linearly with the number of elements in the array. For example if number of elements is doubled then, on average, the search would take twice as long) is rarely used practically because other search algorithms such as the binary search algorithm and hash tables allow significantly faster searching comparison to linear search.

Binary Search

http://studytipsandtricks.blogspot.in/2012/08/explanation-of-local-and-global.html

Local variables:

Variable whose existence is known only to the main program or functions are called local variables. Local variables are declared with in the main program or a function.

Global variables:

Variables whose existence is known to the both main () as well as other functions are called global variables. Global variables are declared outside the main () and other functions.

The following Program illustrates the concept of both local as well as global variables.

```
#include<stdio.h>
// Global variables
int A;
int B;
int Add ()
{
        return A + B;
}
int main ()
{
        int answer; // Local variable
        A = 5;
        B = 7;
        answer = Add ();
        printf ("%d\n",answer);
        return 0;
}
```

Consider the following two definitions of main ().

```
intmain ()
{
/* */
return0;
}
```

Run on IDE
and

```
intmain (void)
{
/* */
return0;
}
```

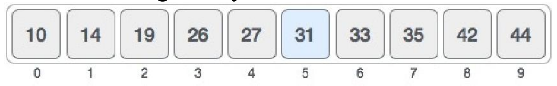
Run on IDE
What is the difference?
In C++, there is no difference, both are same.
Both definitions work in C also, but the second definition with void is considered technically better as it clearly specifies that main can only be called without any parameter.

```
#include <stdio.h>
int main () {
char greeting [6] = {'H', 'e', 'l', 'l', 'o', '\0'};
printf ("Greeting message: %s\n", greeting );
return 0;
}
#include <stdio.h>
#define LENGTH 10
#define WIDTH 5
#define NEWLINE '\n'
int main () {
int area;
area = LENGTH * WIDTH;
printf ("value of area: %d", area);
printf ("%c", NEWLINE);
return 0;
}
#include<stdio.h>
int main (){
constint LENGTH =10;
constint WIDTH =5;
constchar NEWLINE ='\n';
int area;
area = LENGTH * WIDTH;
printf ("value of area: %d", area);
printf ("%c", NEWLINE);
return0;
}
```

Note that it is a good programming practice to define constants in CAPITALS.

Binary Search is applied on the sorted array or list. In binary search, we first compare the value with the elements in the middle position of the array. If the value is matched, then we return the value. If the value is less than the middle element, then it must lie in the lower half of the array and if it's greater than the element then it must lie in the upper half of the array. We repeat this procedure on the lower (or upper) half of the array. Binary Search is useful when there are large numbers of elements in an array.

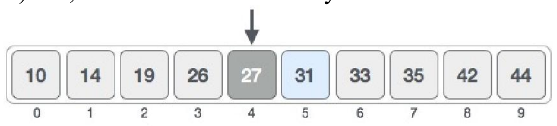
We shall learn the process of binary search with a pictorial example. The following is our sorted array and let us assume that we need to search the location of value 31 using binary search.



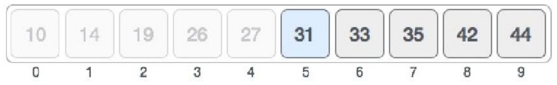
First, we shall determine half of the array by using this formula

mid = low + (high - low) / 2

Here it is, 0 + (9 - 0) / 2 = 4 (integer value of 4.5). So, 4 is the mid of the array.



Now we compare the value stored at location 4, with the value being searched, i.e. 31. We find that the value at location 4 is 27, which is not a match. As the value is greater than 27 and we have a sorted array, so we also know that the target value must be in the upper portion of the array.

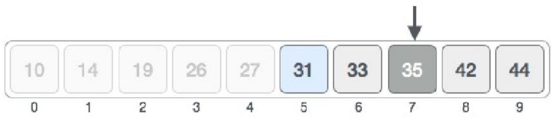


We change our low to mid + 1 and find the new mid value again.

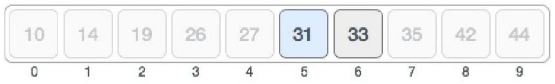
low = mid + 1

mid = low + (high - low) / 2

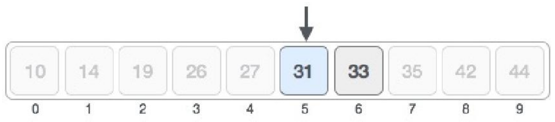
Our new mid is 7 now. We compare the value stored at location 7 with our target value 31.



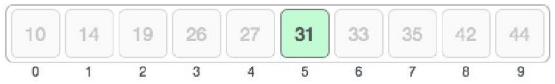
The value stored at location 7 is not a match, rather it is more than what we are looking for. So, the value must be in the lower part from this location.



Hence, we calculate the mid again. This time it is 5.



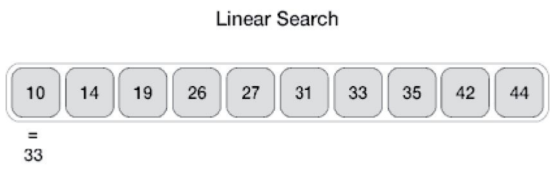
We compare the value stored at location 5 with our target value. We find that it is a match.



We conclude that the target value 31 is stored at location 5.

Binary search tree

Consider a array:



2
1
3
5
9
12
18
15
19
17
12 = root
Left subtree (lesser) < root
Right subtree (greater) > root

--- //

Program 3.3

C program to print whether the given number is positive or negative

```
#include<stdio.h>
int main ()
{
int a;
a = -35;
if (a>0)
{
printf ("number is positive");
}
else
{
printf (" number entered is negative");
}
return 0;
}
```

The output on the screen:

```
number entered is negative
```

Since a = -35. Therefore:

a is less than 0 i.e., a < 0 because any negative number is always less than zero.

The statement

```
{
printf ("number is negative");
}
```

is executed to print the output:

number entered is negative

Program 3.4

C program to print the sum of the first 10 digits using for loop statement

```
#include<stdio.h>
int main ()
{
int i, sum = 0;
for ( i=1; i<=10; i++)
sum = sum + i;
printf ("sum of the first 10 digits =%d", sum);
return 0;
}
```

The output on the screen:

```
sum of the first 10 digits = 55
```

How the sum of the first 10 digits = 55 is outputted on the screen through the for Loop statement?

i=1 (sum = 0 because the sum is initialized to 0 in the statement int i, sum = 0;)

Is i<=10 true?

Yes, do this

sum = sum + i = 0 +1 =1

Now,

i=2 (sum = 1)

Is i<=10 true?

Yes, do this

sum = sum + i = 1 +2 =3

Now,

i=3 (sum = 3)

Is i<=10 true?

Yes, do this

sum = sum + i = 3 +3 = 6

Now,

i=4 (sum = 6)
Is i<=10 true?
Yes, do this
sum = sum + i = 6 + 4= 10
Now,
i=5 (sum = 10)
Is i<=10 true?
Yes, do this
sum = sum + i = 10 + 5= 15
Now,
i=6 (sum = 15)
Is i<=10 true?
Yes, do this
sum = sum + i = 15 + 6 = 21
Now,
i=7 (sum = 21)
Is i<=10 true?
Yes, do this
sum = sum + i = 21 + 7 = 28
Now,
i=8 (sum = 28)
Is i<=10 true?
Yes, do this
sum = sum + i = 28 + 8 = 36
Now,
i=9 (sum = 36)
Is i<=10 true?
Yes, do this
sum = sum + i = 36 + 9 = 45
Now,
i=10 (sum = 45)
Is i<=10 true?
Yes, do this
sum = sum + i = 45 + 10 = 55
stops because the condition i<=10 is achieved
The statement:

```
printf ("sum of the first 10 digits =%d", sum);
```

is executed to print the output:

```
sum of the first 10 digits = 55
```

If the statement:
int i, sum = 0;
is replaced by int i, sum = 1;
Then the output on the screen is:

```
sum of the first10 digits = 56
```

What will be the output if the for loop statement for (i =1; i<=10; i++) is replaced by the statement for (i =2; i<10; i++)?

Answer: sum of 10 digits = 44

If the statement int i, sum, sum = 0; is written instead of int i, sum = 0;

Then the compilation error message will be displayed on the screen (stating that sum is twice declared).

If the for loop is ended with a semicolon i.e.,
for (i=1; i<=10; i++);

Then the compilation error will be displayed on the console screen.

Note:
sum = sum + a; is the same as sum + = a;
sub = sub- a; is the same as sub - = a;
product = product* a; is the same as product * = a;
div = div / a; is the same as div /= a;
a = a% b; is the same as a % = b;

Program 3.5

C program to print the average of the first 10 numbers using for loop statement

```
#include<stdio.h>
int main ()
{
int i, avg, sum = 0;
for ( i=1; i<=10; i++)
sum = sum + i;
avg = sum/10;
printf ("sum of the first 10 numbers =%d", sum);
printf ("average of the first 10 numbers =%d",
avg);
return 0;
}
```

The output on the screen:

```
sum of the first 10 numbers = 55
average of the first 10 numbers = 5
```

The average of the first10 numbers = 55/10 = 5.5 not 5. But the output on the screen is:

```
average of the first 10 numbers = 5
```

because int is used instead of float.
If the data type float is used i.e.,

```
#include<stdio.h>
int main ()
{
float i, avg, sum = 0;
for ( i=1; i<=10; i++)
sum = sum + i;
avg = sum/10;
printf ("sum of the first10 numbers =%f", sum);
printf ("average of the first10 numbers = %f",
avg);
return 0;
}
```

The output on the screen:

```
sum of the first10 numbers = 55
average of the first 10 numbers = 5.5
```

Program 3.6

C program to print the product of the first 10 digits using for loop statement

```
#include<stdio.h>
int main ()
{
int i, product = 1;
for ( i=1; i<=10; i++)
product = product * i;
```

printf ("the product of the first 10 digits =%d", product);

return 0;

}

The output on the screen:

the product of the first 10 digits = 3628800

How the product of the first 10 digits = 3628800 is outputted on the screen through the for Loop statement?

i=1 (product = 1 because the product is initialized to 1 in the statement int i, product = 1;)

Is i<=10 true?

Yes, do this

product = product * i = 1 * 1 =1

Now,

i=2 (product = 1)

Is i<=10 true?

Yes, do this

product = product * i = 1 * 2 = 2

Now,

i=3 (product = 2)

Is i<=10 true?

Yes, do this

product = product * i = 2 * 3 = 6

Now,

i=4 (product = 6)

Is i<=10 true?

Yes, do this

product = product * i = 6 * 4 = 24

Now,

i=5 (product =24)

Is i<=10 true?

Yes, do this

product = product * i = 24 * 5 =120

Now,

i=6 (product =120)

Is i<=10 true?

Yes, do this

product = product * i = 120 * 6 = 720

Now,

i=7 (product =720)

Is i<=10 true?

Yes, do this

product = product * i = 720 * 7 = 5040

Now,

i=8 (product =5040)

Is i<=10 true?

Yes, do this

product = product * i = 5040 * 8 = 40320

Now,

i=9 (product = 40320)

Is i<=10 true?

Yes, do this

product = product * i = 40320 * 9 = 362880

Now,

i=10 (product = 362880)

Is i<=10 true?

Yes, do this

product = product * i = 362880 * 10 = 3628800

stops because the condition i<=10 is achieved.

The statement:

printf ("the product of the first 10 digits =%d", product); is executed to display the output:

the product of the first 10 digits = 3628800

If the statement int i, product = 1; is replaced by int i, product = 0;

Then the output on the screen is:

the product of the first 10 digits = 0

If the statement for (i=1; i<=10; i++) is replaced by for (i=5; i<=8; i++)

Then the output on the screen is:

the product of the first 10 digits = 1680

Program 3.7

C Program to print the table of a number using the for loop statement

```
#include<stdio.h>
int main ()
{
int n, i;
printf ("Enter any number:");
scanf ("%d",     &   n);
for ( i=1; i<=5; i++)
printf ("%d * %d = %d\n", n, i, n*i);
return 0;
}
```

The output on the screen:

```
Enter any number:
```

If you enter the number 2 (i.e., n=2)

```
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
```

will be outputted on the screen.

How the execution takes its Way through the for Loop statement

Since you entered the number 2, therefore: n=2.

i=1

Is i<=5 true?

Yes, print this

2 * 1 = 2

using the statement printf ("%d * %d = %d\n", n, i, n*i);

Now,

i=2

Is i<=5 true?

Yes, print this

2 * 2 = 4

using the statement printf ("%d * %d = %d\n", n, i, n*i);

Now,

i=3

Is i<=5 true?

Yes, print this

2 * 3 = 6

using the statement printf ("%d * %d = %d\n", n, i, n*i);

Now,

i=4

Is i<=5 true?

Yes, print this

2 * 4 = 8

using the statement printf ("%d * %d = %d\n", n, i, n*i);

Now,

i=5

Is i<=5 true?

Yes, print this

2 * 5 = 10

using the statement printf ("%d * %d = %d\n", n, i, n*i);

stop Now because the condition i <=5 is achieved.

If the symbol * is replaced by +

i.e.,

```
#include<stdio.h>
int main ()
{
int n, a;
printf ("Enter any number:");
scanf ("%d",     &   n);
for ( i=1; i<=5; i++)
printf ("%d + %d = %d\n", n, i, n+ i);
return 0;
}
```

Then the output on the screen is:

```
Enter any number:
If you enter the number 2 (i.e., n=2)
2 + 1 = 3
2 + 2 = 4
2 + 3 = 5
2 + 4 = 6
2 + 5 = 7
```

will be outputted on the screen.

Program 3.8

C program:

If you enter a character M

Output must be: ch = M

```
#include<stdio.h>
int main ()
{
char M;
printf ("Enter any character:");
scanf ("%c",     &   M);
printf ("ch=%c", M);
return 0;
}
```

The output on the screen:

Enter any character:

If you enter the character M

ch = M will be outputted on the screen.

Note:

getchar () function is simplified version of the scanf function

If we replace the statement scanf ("%c",   & M); by the statement:

M = getchar ();

i.e.,

```
#include<stdio.h>
int main ()
{
char M;
printf ("Enter any character:");
M = getchar ();
printf ("ch=%c", M);
return 0;
}
```

There will be no change in the output on the screen i.e., The output on the screen is:

Enter any character:

If you enter the character K

ch = K will be outputted on the screen.

putchar () function is simplified version of the printf function

If we replace the statement printf ("ch=%c", M);by the statement:

putchar (M); i.e.,

```
#include<stdio.h>
int main ()
{
char M;
printf ("Enter any character:");
scanf ("%c",     &   M);
putchar (M);
return 0;
}
```

Then there will be no change in the output on the screen i.e., The output on the screen is:

Enter any character:

If you enter the character M

M will be outputted on the screen.

If you replace the statement scanf ("%c", & M); by the statement:

M = getchar ();

and the statement printf ("ch=%c", M);by the statement:

putchar (M); i.e.,

```
#include<stdio.h>
int main ()
{
char M;
printf ("Enter any character:");
M = getchar ();
putchar (M);
return 0;
}
```

The output on the screen:

```
Enter any character:
```

If you enter the character S

S will be outputted on the screen.

Program 3.9

C program to print the first 5 numbers starting from one together with their squares.

```
#include<stdio.h>
int main ()
{
int i;
for ( i=1; i<=5; i++)
printf ("number=%d its square=%d\n", i, i*i);
return 0;
}
```

The output on the screen:

```
number=1 its square=1
number=2 its square=4
number=3 its square=9
number=4 its square=16
number=5 its square=25
```

How the execution takes its way through the for loop statement

i=1

Is i<=5 true?

Yes, print this

number=1 its square=1

using the statement printf ("number=%d its square=%d\n", i, i*i);

Now,

i=2

Is i<=5 true?

Yes, print this

number=2 its square=4

using the statement printf ("number=%d its square=%d\n", i, i*i);

Now,

i=3

Is i<=5 true?

Yes, print this

number=3 its square=9

using the statement printf ("number=%d its square=%d\n", i, i*i);

Now,

i=4

Is i<=5 true?

Yes, print this

number=4 its square=16

using the statement printf ("number=%d its square=%d\n", i, i*i);

Now,

i=5

Is i<=5 true?

Yes, print this

number=5 its square=25

using the statement printf ("number=%d its square=%d\n", i, i*i);

stop Now because the condition (i<=5) is achieved.

Note:

If the statement

printf ("number=%d its square=%d\n", i, i*i);

is replaced by the statement:

printf ("\n number=%d/t its square=%d", i, i*i);

Then the output on the screen is:

```
number=1       its square=1
number=2       its square=4
number=3       its square=9
number=4       its square=16
number=5       its square=25
```

tab /t is included because to leave space between

number=1and its square=1

Suppose printf ("number=%d its square=%d", a, a*a); is replaced by the statement:

printf ("number=%d\n its square=%d\n", a, a*a);

The output on the screen is:

```
number=1
its square=1
number=2
its square=4
number=3
its square=9
number=4
its square=16
number=5
its square=25
```

And if you replace the printf statement:

printf ("number=%d its square=%d", a, a*a); by the statement:

printf ("number=%d\n, its square=%d\n", a, a*a);

i.e., if you place variable separator (i.e., comma) between number=%d\n and its square=%d\n

Then the compilation error will be displayed on the screen.

Write a program to print the first 10 numbers starting from one together with their squares and cubes?

Answer:

```
#include<stdio.h>
int main ()
{
int i;
for ( i=1; i<=10; i++)
printf ("number=%d its square=%d its cube=%d\n", i, i*i, i*i*i);
return 0;
}
```

Program 4.0

C program to print the sum of two numbers using pointers

If we create an integer variable x by declaring the statement:

int x;

within the body of the main function int main () -- this variable is stored in the computer memory i.e., this variable occupies a specific location in the space of computer memory.

And this integer variable x is assigned an address (i.e., & x) to locate its position in the computer memory (like a house in the street is assigned an address to locate its position in the street).

Pointers are the variables that represent the address of x in the computer memory i.e., p = & x, where & x imply the address of x in the computer memory and p is the pointer variable (which is the variable that represent the address of x in the computer memory).

And further if you assign a value to the variable x by declaring the statement:

x=1;

within the body of the main functionthis value is stored in the address of x in the computer memory. "*" denote pointer operator and *p denote the pointer

(which represent the value stored in the address of x in the computer memory).

C program to print the address of x and the value assigned to x

```
#include <stdio.h>
int main ()
{
int x, *p;
x = 1;
p = & x;
printf ("The address of the variable x =%d", p);
printf ("The value of the variable x =%d", *p);
return 0;
}
```

The output on the screen:

```
The address of the variable x = 0x7fffc60478a4
The value of the variable x = 1
```

Since p = & x:

*p= * & x

The value of the variable x = 1 because you have assigned a value to the variable x by declaring the statement:

x=1;

within the body of the main function.

If the statements:

```
printf ("The address of the variable x =%d", p);
printf ("The value of the variable x =%d", *p);
```

are replaced by the statement:

```
printf ("The address of the variable x =%d and its value =%d", p,*p);
```

i.e.,

```
#include <stdio.h>
int main ()
{
int x, *p;
x=1;
p = & x;
printf ("The address of the variable x =%d and its value =%d", p,*p);
return 0;
}
```

Then the output on the screen is:

```
The address of the variable x = 0x7fffc60478a4and its value = 1
```

```
#include <stdio.h>
int main ()
{
int x, y, *p, *q, sum;
printf ("Enter any number:");
scanf ("%d", & x);
printf ("Enter any number:");
scanf ("%d", & y);
p = & x;
q = & y;
sum = *p + *q;
printf ("Sum of entered numbers = %d\n", sum);
return 0;
}
```

The output on the screen:

Enter any number:

If you enter the number 2

Enter any number:

If you enter the number 3

Sum of entered numbers = 5 will be outputted on the screen.

Since pointer *p imply the value assigned to the variable x (i.e., 2) through the keyboard and the pointer *q imply the value assigned to the variable y (i.e., 3) through the keyboard. Therefore:

sum = *p + *q = 2 + 3 = 5 (which will be outputted on the screen)

C program to print the product, subtraction and division of two numbers using pointers

```
#include <stdio.h>
int main ()
{
int x, y, *p, *q, product, subtract, div;
printf ("Enter any number:");
scanf ("%d", & x);
printf ("Enter any number:");
scanf ("%d", & y);
p = & x;
q = & y;
product = *p * *q;
subtract = *p - *q;
div= *p / *q;
printf ("product of entered numbers = %d\n",
product);
printf ("subtract of entered numbers = %d\n", subtract);
printf ("division of entered numbers = %d\n", div);
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter the number 4
Enter any number:
If you enter the number 2
product of entered numbers = 8
subtract of entered numbers = 2
division of entered numbers = 2
```

will be outputted on the screen.

C program to find the greatest of two numbers using pointers

```
#include<stdio.h>
int main ()
{
int x, y, *p, *q;
printf ("Enter any integer:");
scanf ("%d",    &  x);
printf ("Enter any integer:");
scanf ("%d",    &  y);
p =   &  x;
q =   &  y;
if (*p>*q)
{
printf ("x is greater than y");
}
if (*q>*p)
{
printf ("y is greater than x");
}
return 0;
}
```

The output on the screen:

```
Enter any integer:
If you enter the integer 10
Enter any integer:
If you enter the integer 16
```

y is greater than x will be outputted on the screen.

What is the output of the following programs:

i)

```
#include <stdio.h>
int main ()
{
int x;
x=12;
printf ("per = %d%", x);
return 0;
}
```

Answer:

```
per=12
```

ii)

```
#include <stdio.h>
int main ()
{
int x, t, c;
x =12;
t = 2;
c = x/t;
printf ("velocity = %d m/s", c);
return 0;
}
```

Answer:

```
velocity = 6 m/s
```

Program 4.1

C program to print the sum of two numbers using functions

```
#include<stdio.h>
int addition ();
int main ()
{
int answer;
answer = addition ();
printf ("The sum of two numbers is: %d\n",answer);
return 0;
}
int addition ()
{
int x, y;
printf ("Enter any integer:");
scanf ("%d",    &  x);
printf ("Enter any integer:");
scanf ("%d",    &  y);
return x+y;
}
```

The output on the screen:

```
Enter any integer:
If you enter the integer 3
Enter any integer:
If you enter the integer 5
```

sum of two numbers = 8 will be displayed on the screen.

int addition (); // the statement implies function declaration

int means integer and int addition () implies: addition () should return integer value.

int addition ()// implies: the function to add the entered values (i.e., 3 and 5) and return the result (i.e., 3 + 5 i.e., 8) to the statement:

printf ("sum of two numbers = %d", answer); to make provision to display the output:

sum of two numbers = 8

```
{
int x, y;
printf ("Enter any integer:");
scanf ("%d",    &   x);
printf ("Enter any integer:");
scanf ("%d",    &   y);
return x+y;
} // implies: the body of the function int addition ()
answer = addition (); // implies: the function call
```

i.e., this statement calls the function:

addition ()

to add the entered values (i.e., 3 and 5) and return the result (i.e., 3 + 5 i.e., 8)

to the statement:

printf ("sum of two numbers = %d", answer);

to make provision to display the output:

sum of two numbers = 8

on the screen.

In the statement:

printf ("sum of two numbers=%d", answer);

the format string %d indicates that the value to be displayed at that point in the string i.e., after the statement:

sum of two numbers =

needs to be taken from the result returned by the function int addition ().

C program to print the product of two numbers using functions

```
#include<stdio.h>
int multiplication ();
int main ()
{
int answer;
answer = multiplication ();
printf ("The product of two numbers is: %d\n",answer);
return 0;
}
int multiplication ()
{
int x, y;
printf ("Enter any integer:");
scanf ("%d",    &   x);
printf ("Enter any integer:");
scanf ("%d",    &   y);
return x*y;
}
```

The output on the screen:

Enter any integer:

If you enter the integer 3

Enter any integer:

If you enter the integer 5

product of two numbers = 15 will be outputted on the screen.

C program to print the greatest of two numbers using functions

```
#include<stdio.h>
int largest ();
int main ()
{
int answer;
answer = largest ();
printf ("The largest of two numbers is: %d\n",answer);
return 0;
}
int largest ()
{
int x, y;
printf ("Enter any integer:");
scanf ("%d",    &   x);
printf ("Enter any integer:");
scanf ("%d",    &   y);
if (x>y)
return x;
if (y>x)
return y;
}
```

The output on the screen:

Enter any integer:

If you enter the integer 3

Enter any integer:

If you enter the integer 5

largest of two numbers= 5 will be outputted on the screen.

C program to print the greatest of three numbers using functions

```
#include<stdio.h>
int largest ();
int main ()
{
int answer;
answer = largest ();
printf ("largest of three numbers=%d", answer);
return 0;
}
int largest ()
{
int x, y, z;
printf ("Enter any integer:");
scanf ("%d",    &   x);
printf ("Enter any integer:");
scanf ("%d",    &   y);
printf ("Enter any integer:");
scanf ("%d",    &   z);
if (x>y    &    &   x>z)
return x;
if (y>x    &    &   y > z)
return y;
if (z>x    &    &   z>y)
return z;
}
```

The output on the screen:

Enter any integer:

```
If you enter the integer 3
Enter any integer:
If you enter the integer 5
Enter any integer:
If you enter the integer 10
largest of three numbers = 10 will be outputted on the screen.
```

C program to print the square of the number using functions

```
#include<stdio.h>
int square ();
int main ()
{
int answer;
answer = square ();
printf ("square of the given number=%d", answer);
}
int square ()
{
int x;
printf ("Enter any integer:");
scanf ("%d",    &   x);
return x*x;
}
```

The output on the screen is:

```
Enter any integer:
If you enter an integer 5
square of the number = 25 will be outputted on the screen.
```

What is the output of the following program:

```
#include<stdio.h>
int main ()
{
int x;
x=6;
printf ("The address of x = %d",     &   x);
return 0;
}
```

Answer:

```
The address of x = -604171156
```

Program 4.2

Switch (case) allows to make decision from the number of choices i.e., from the number of cases

For example:

```
#include<stdio.h>
int main ()
{
char ch;
printf ("Enter any character:");
scanf ("%c",    &   ch);
switch (ch)
{
case 'R':
printf ("Red");
break;
case 'W':
printf ("White");
break;
case 'Y':
printf ("Yellow");
break;
case 'G':
printf ("Green");
break;
default:
printf ("Error");
break;
}
return 0;
}
```

The output on the screen:

```
Enter any character:
If you enter a character R
Red will be outputted on the screen.
```

switch (ch) allow to make decision from the number of choices i.e., from the number of cases

```
case 'R':
case 'W':
case 'Y':
case 'G':
```

Since we have entered the character R (which corresponds to case 'R':)

The statement

```
printf ("Red");
```

is executed to display the output:

```
Red
```

on the screen.

Suppose you enter a character K

Then the output on the screen is:

```
Error
```

(Entered character K does not correspond to any of the cases:

```
case 'R':
case 'W':
case 'Y':
case 'G':
```

Therefore the statement:

```
printf ("Error");
```

is executed to display the output:

```
Error
```

on the screen).

If the statements:

```
case 'R':
printf ("Red");
break;
case 'W':
printf ("White");
break;
case 'Y':
printf ("Yellow");
break;
```

```
case 'G':
printf ("Green");
break;
default:
printf ("Error");
break;
```

are replaced by the statements:

```
case 'R':
printf ("Red");
case 'W':
printf ("White");
case 'Y':
printf ("Yellow");
break;
case 'G':
printf ("Green");
break;
default:
printf ("Error");
break;
```

Then the output on the screen is:

```
Red
White
Yellow
```

i.e., the output will be printed till yellow even though you have entered the character R.

Program 4.3

C program to print the output:

```
Element [0] = 16
Element [1] = 18
Element [2] = 20
Element [3] = 25
Element [4] = 36
```

using arrays:

```
#include<stdio.h>
int main ()
{
int i;
int num [5] = {16, 18, 20, 25, 36};
for (i=0; i<5; i++)
printf ("\n Element [%d] = %d", i, num [i]);
return 0;
}
```

The output on the screen:

```
Element [0] = 16
Element [1] = 18
Element [2] = 20
Element [3] = 25
Element [4] = 36
```

The statement:

int num [5] = {16, 18, 20, 25, 36};

imply that we are creating an integer array (and the name of array is num) consisting of 5 values (i.e., 16, 18, 20, 25, 36) of the same data type int.

The number of values between the braces { } cannot be larger than the number of values that we declare for the array between square brackets [ ].

There are 5 integers i.e., 16, 18, 20, 25, 36 within the braces { }, so 5 is written within the square brackets [ ].

If there were 6 integers i.e., 16, 18, 20, 25, 36, 42 within the braces { }, then 6 must be written within the square brackets [ ].

Note: With the declaration int num [5], computer creates 5 memory cells with name num [0], num [1], num [2], num [3], num [4].

And since:

int num [5] = {16, 18, 20, 25, 36};

the values 16, 18, 20, 25, 36 are stored in num [0], num [1], num [2], num [3], num [4] respectively.

How the execution takes its way through the for loop statement

i=0

Is i<5 true?

Yes, print this

Element [0] = 16

using the statement:

printf ("\n Element [%d] = %d", i, num [i])

format string %d in the square brackets indicates that the value to be displayed at that point in the string i.e., with the square brackets [ ] needs to be taken from a variable (which is i i.e., i=0) and the format string %d after the statement (\n Element [%d] =) indicates that the value to be displayed at that point in the string i.e., after the statement (\n Element [%d] =) needs to be taken from a variable (which is stored in num [i] i.e., num [0] i.e., 16).

Now,

i=1

Is i<5 true?

Yes, print this

Element [1] = 18

using the statement:

printf ("\n Element [%d] = %d", i, num [i])

format string %d in the square brackets indicates that the value to be displayed at that point in the string i.e., with the square brackets [ ] needs to be taken from a variable (which is i i.e., i=1) and the format string %d after the statement (\n Element [%d] =) indicates that the value to be displayed at that point in the string i.e., after the statement (\n Element [%d] =) needs to be taken from a variable (which is stored in num [i] i.e., num [1] i.e., 18).

Now,

i=2

Is i<5 true?

Yes, print this

Element [2] = 20

using the statement:

printf ("\n Element [%d] = %d", i, num [i])

format string %d in the square brackets indicates that the value to be displayed at that point in the string i.e., with the square brackets [ ] needs to be taken from a variable (which is i i.e., i=2) and the format string %d after the statement (\n Element [%d] =) indicates that the value to be displayed at that point in the string i.e., after the statement (\n Element [%d] =) needs to be taken from a variable (which is stored in num [i] i.e., num [2] i.e., 20).

Now,
i=3
Is i<5 true?
Yes, print this
Element [3] = 25
using the statement:
printf ("\n Element [%d] = %d", i, num [i])

format string %d in the square brackets indicates that the value to be displayed at that point in the string i.e., with the square brackets [ ] needs to be taken from a variable (which is i i.e., i=3) and the format string %d after the statement (\n Element [%d] =) indicates that the value to be displayed at that point in the string i.e., after the statement (\n Element [%d] =) needs to be taken from a variable (which is stored in num [i] i.e., num [3] i.e., 25).

Now,
i=4
Is i<5 true?
Yes, print this
Element [4] = 36
using the statement:
printf ("\n Element [%d] = %d", i, num [i])
Stop because the condition i<5 is achieved.

format string %d in the square brackets indicates that the value to be displayed at that point in the string i.e., with the square brackets [ ] needs to be taken from a variable (which is i i.e., i=4) and the format string %d after the statement (\n Element [%d] =) indicates that the value to be displayed at that point in the string i.e., after the statement (\n Element [%d] =) needs to be taken from a variable (which is stored in num [i] i.e., num [4] i.e., 36).

Suppose the statement:
printf ("\n Element [%d] = %d", i, num [i]); is replaced by the statement:
printf ("\n Element [%d] = %d", i, num [0]);
Then the output on the screen:
```
Element [0] = 16
Element [1] = 16
Element [2] = 16
Element [3] = 16
Element [4] = 16
```
Suppose the statement:
printf ("\n Element [%d] = %d", i, num [i]); is replaced by the statement:
printf ("\n Element [%d] = %d", i, num [1]);
The output on the screen:
```
Element [0] = 18
Element [1] = 18
Element [2] = 18
Element [3] = 18
Element [4] = 18
```
Suppose the statement:
printf ("\n Element [%d] = %d", i, num [i]); is replaced by the statement:
printf ("\n Element [%d] = %d", i, num [2]);
The output on the screen:
```
Element [0] = 20
Element [1] = 20
Element [2] = 20
Element [3] = 20
Element [4] = 20
```
Suppose the statement:
printf ("\n Element [%d] = %d", i, num [i]); is replaced by the statement:
printf ("\n Element [%d] = %d", i, num [3]);
The output on the screen:
```
Element [0] = 25
Element [1] = 25
Element [2] = 25
Element [3] = 25
Element [4] = 25
```
Suppose the statement:
printf ("\n Element [%d] = %d", i, num [i]); is replaced by the statement:
printf ("\n Element [%d] = %d", i, num [4]);
The output on the screen:
```
Element [0] = 36
Element [1] = 36
Element [2] = 36
Element [3] = 36
Element [4] = 36
```
If the condition:
i<5
is replaced by the condition:
i<=5
Then the output on the screen is:
```
Element [0] = 16
Element [1] = 18
Element [2] = 20
Element [3] = 25
Element [4] = 36
Element [5] = 3656
```
3656 is the number stored in the memory i.e., any number stored in the memory will be displayed.

If the statement:
int num [5] = {16, 18, 20, 25, 36}; is replaced by the statement:
int num [i] = {16, 18, 20, 25, 36};

Then the compilation will be displayed on the screen because there are 5 elements within the braces {} not i elements.

Note:

C program to print the sum of the elements in array.

```
#include<stdio.h>
int main ()
{
int i, sum = 0;
int num [5] = {16, 18, 20, 25, 36};
for (i=0; i<5; i++)
sum = sum + num [i];
printf ("Sum of the Elements in the array = %d", sum);
return 0;
}
```

The output on the screen:

```
Sum of the Elements in the array = 115
```

i.e., 16 + 18 + 20 + 25 + 36 = 115

How the Execution takes its way through the for loop statement

i=0 (sum = 0)
Is i<5 true?
Yes, do this
sum = sum + num [i] = sum + num [0] = 0 +16 =16
Now,
i=1 (sum = 16)
Is i<5 true?
Yes, do this
sum = sum + num [i] = sum + num [1] = 16 +18 =34
Now,
i=2 (sum = 34)
Is i<5 true?
Yes, do this
sum = sum + num [i] = sum + num [2] = 34 +20 =54
Now,
i=3 (sum = 54)
Is i<5 true?
Yes, do this
sum = sum + num [i] = sum + num [3] = 54 +25 =79
Now,
i=5 (sum = 79)
Is i<5 true?
Yes, do this
sum = sum + num [i] = sum + num [5] = 79 + 36 =115
stop because the condition i<5 is achieved

The statement:
printf ("Sum of the Elements in the array = %d", sum); is executed to display the output:
Sum of the Elements in the array = 115
on the screen.

If the statement:
int i, sum = 0;
is replaced by int i, sum = 1;
Then The output on the screen:
Sum of the Elements in the array = 116

C program to print the average of the elements in array

```
#include<stdio.h>
int main ()
{
int i, avg, sum = 0;
int num [5] = {16, 18, 20, 25, 36};
for (i=0; i<5; i++)
sum = sum + num [i];
avg = sum/5;
printf ("Sum of the Elements in the array = %d", sum);
printf ("average of the elements in the array= %d", avg);
return 0;
}
```

The output on the screen:

```
Sum of the Elements in the array = 115
average of the elements in the array = 23
```

Write a program to print:

```
Einstein [0] = E
Einstein [1] = I
Einstein [2] = N
Einstein [3] = S
Einstein [4] = T
Einstein [5] = E
Einstein [6] = I
Einstein [7] = N
```

using arrays

Answer:

```
#include<stdio.h>
int main ()
{
int i;
char name [8] = {' E', ' I', ' N', ' S', ' T ', ' E', ' I', ' N'};
for (i=0; i<8; i++)
printf ("\n Element [%d] = %c", i, name [i]);
return 0;
}
```

Note:
If the format string %d is used instead of %c i.e., if the statement:
printf ("\n Element [%d] = %c", name [i], name [i]); is written instead of the statement:
printf ("\n Element [%c] = %c", name [i], name [i]);

Then the output on the screen is:

```
Element [69] = E
Element [73] = I
Element [78] = N
Element [83] = S
Element [84] = T
Element [69] = E
Element [73] = I
Element [78] = N
```

What will be the output of the following programs?

i)

```
#include <stdio.h>
#include <math.h>
int main ()
{
printf ("%f", cbrt (27));
return 0;
}
```

Answer:

```
3.000
```

ii)

```
#include <stdio.h>
int main ()
{
char i;
char body [4] = {'b', 'o', 'd', 'y'};
for (i=0; i<4; i++)
printf ("\n body [%c] = %c", body [i], body [i]);
return 0;
}
```

Answer:

```
body [b] = b
body [o] = o
body [d] = d
body [y] = y
```

iii)

```
#include <stdio.h>
#include <malloc.h>
int main ()
{
int x=2;
printf ("%d", malloc ( 200*sizeof (x)));
return 0;
}
```

Answer:

```
8183824
```

What is the mistake in the following program:

```
#include<stdio.h>
int main ()
{
int i;
int num [] = {16, 18, 20, 25, 36};
for (i=0; i<5; i++)
printf ("\n Element [%d] = %d", i, num [i]);
return 0;
}
```

Answer: There is no mistake in the above program. The output on the screen is:

```
Element [0] = 16
Element [1] = 18
Element [2] = 20
Element [3] = 25
Element [4] = 36
```

Program 4.3

C program to print the output:

```
Name of the book = B
Price of the book = 135.00
Number of pages = 300
Edition = 8
```

using structures

```
#include<stdio.h>
int main ()
{
struct book {
char name;
float price;
int pages;
int edition;
};
struct book b1;
b1.name = 'B';
b1.price = 135.00;
b1.pages = 300;
b1.edition = 8;
printf ("\n Name of the book = %c", b1.name);
printf ("\n Price of the book = %f", b1.price);
printf ("\n Number of pages = %d", b1.pages);
printf ("\n Edition of the book = %d", b1.edition);
return 0;
}
```

The output on the screen:

```
Name of the book = B
Price of the book = 135.00
Number of pages = 300
Edition of the book = 8
```

The statement:

```
struct book {
char name;
float price;
int pages;
int edition;
};
```

imply the structure definition i.e., we are defining a structure (and the data type name of the structure is book) and it consists of elements:

name (which is of data type char), price (which is of data type float), pages (which is of data type int), edition (which is of data type int) which are placed within the body of the structure.

The statement:

```
struct book b1;
```

imply the structure variable declaration (where b1 denote the structure variable)

Why structure variable b1 is declared or defined?

In order to assign the values to the elements within the body of the structure, each element must be linked with structure variable with dot operator or period operator or member accessibility operator.

For example: name is the element which must be linked with structure variable b1 with dot operator to assign a value B to the element name.

format string %c (corresponding to the data type char) in the statement:

printf ("\n Name of the book = %c", b1.name);

indicates that the value to be displayed at that point in the string i.e., after the statement (\n Name of the book =) needs to be taken from b1.name.

The statement:

printf ("\n Name of the book = %c", b1.name);

make provision to print the output:

Name of the book = B

on the screen.

format string %f (corresponding to the data type float) in the statement:

printf ("\n Price of the book = %f", b1.price);

indicates that the value to be displayed at that point in the string i.e., after the statement (\n Price of the book =) needs to be taken from b1.price.

The statement:

printf ("\n Price of the book = %f", b1.price);

make provision to print the output:

Price of the book = 135.00

on the screen.

format string %d (corresponding to the data type int) in the statement:

printf ("\n Number of pages = %d", b1.pages);

indicates that the value to be displayed at that point in the string i.e., after the statement (\n Number of pages =) needs to be taken from b1.pages.

The statement:

printf ("\n Number of pages = %d", b1.pages);

make provision to print the output:

Number of pages = 300

on the screen.

format string %d (corresponding to the data type int) in the statement:

printf ("\n Edition of the book = %d", b1.edition);

indicates that the value to be displayed at that point in the string i.e., after the statement (\n Edition of the book =) needs to be taken from b1.edition.

The statement:

printf ("\n Edition of the book = %d", b1.edition);

make provision to print the output:

Edition of the book = 8

on the screen.

What will be output of the following programs?

A)

```
#include<stdio.h>
struct book {
char name;
float price;
int pages;
int edition;
};
int main ()
{
struct book b1;
b1.name = 'B';
b1.price = 135.00;
b1.pages = 300;
b1.edition = 8;
printf ("\n Name of the book = %c", b1.name);
printf ("\n Price of the book = %f", b1.price);
printf ("\n Number of pages = %d", b1.pages);
printf ("\n Edition of the book = %d", b1.edition);
}
```

Answer:

```
Name of the book = B
Price of the book = 135.000000
Number of pages = 300
Edition of the book = 8
```

B)

```
#include <stdio.h>
int main (){
for (;; ) {
printf ("This loop will run forever.\n");
}
return 0;
}
```

Answer:

```
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever.......... continues
```

C)

```
#include<stdio.h>
int main ()
{
char ch [5];
printf ( "Enter the name: ");
scanf ("%s",     &  ch);
printf ( "the name you entered = %s", ch);
return 0;
}
```

Answer:

Enter the name:

If you enter the name Dennis

the name you entered = Denni will be outputted on the screen.

Instead of Dennis, only Denni will be displayed on the screen because of the statement char ch [5];

The statement:

char ch [5];

make provision only for 5 lettered name to be displayed on the screen.

If the statement:

char ch [5]; is replaced by the statement char ch

[6];

Then the output on the screen is:

Enter the name:

If you enter the name Dennis

the name you entered = Dennis will be outputted on the screen.

Note: %s implies the format specifier for string.

Program 4.4

Continue and break statements:

i)

```
#include <stdio.h>
int main ()
{
int i;
for (i=1; i<=5; i++)
{
if (i==3)
{
continue;
}
printf ("%d\n ", i);
}
return 0;
}
```

Output on the screen:

```
1
2
4
5
```

Note:

i = 1

Is the condition (i<=5) is true?

Yes because i=1

The statement printf ("%d\n ", i); is executed to print the output:

1

Now, the value of i is:

i = 1+1 = 2

Is the condition (i<=5) is true?

Yes because i=2

The statement printf ("%d\n ", i); is executed to print the output:

2

Now, the value of i is:

i = 2+1 = 3

Is the condition (i<=5) is true?

Yes because i=3

The statement printf ("%d\n ", i); is not executed to print the output:

3

Because of the statement:

```
if (i==3)
{
continue;
}
// Execution skips //
```

Now, the value of i is:

i = 3+1 = 4

Is the condition (i<=5) is true?

Yes because i=4

The statement printf ("%d\n ", i); is executed to print the output:

4

Now, the value of i is:

i = 4+1 = 5

Is the condition (i<=5) is true?

Yes because i=5

The statement printf ("%d\n ", i); is executed to print the output:

5

and stop because the condition i<=5 is achieved.

ii)

```
#include <stdio.h>
int main ()
{
int i;
for (i=1; i<=5; i++)
{
if (i==3)
{
break;
}
printf ("%d\n ", i);
}
return 0;
}
```

Output on the screen:

```
1
2
```

Note:

i = 1

Is the condition (i<=5) is true?

Yes because i=1

The statement printf ("%d\n ", i); is executed to print the output:

1

Now, the value of i is:

i = 1+1 = 2

Is the condition (i<=5) is true?

Yes because i=2

The statement printf ("%d\n ", i); is executed to print the output:

2

Now, the value of i is:

i = 2+1 = 3

Is the condition (i<=5) is true?

Yes because i=3

The statement printf ("%d\n ", i); is not executed to print the output:

Because of the statement:

```
if (i==3)
{
```

```
break;
}
```

The for loop:

for (i=1; i<=5; i++)

is immediately terminated (even before the condition i<=5 is achieved) and program execution stops.

//---

The goto statement:

```
#include <stdio.h>
int main ()
{
int i;
for (i=1;i<=5;i++)
{
if (i==3)
{
goto HAI;
}
printf ("\n %d ",i);
}
HAI: printf ("\n Linux");
}
```

Output on the screen:

```
1
2
Linux
```

Note:

i = 1

Is the condition (i<=5) is true?

Yes because i=1

The statement printf ("\n %d ",i); is executed to print the output:

1

Now, the value of i is:

i = 1+1 = 2

Is the condition (i<=5) is true?

Yes because i=2

The statement printf ("\n %d ",i); is executed to print the output:

2

Now, the value of i is:

i = 2+1 = 3

Is the condition (i<=5) is true?

Yes because i=3

The statement printf ("%d\n ", i); is not executed to print the output:

3

Rather

The statement printf ("\n Linux"); is executed to print the output:

Linux

Because of the statement:

```
if (i==3)
{
goto HAI;
}
```

The for loop:

for (i=1; i<=5; i++)

is immediately terminated (even before the condition i<=5 is achieved) and program execution stops.

---//

Program 4.5

C program to convert the upper case letter to lower case letter

```
#include<stdio.h>
int main ()
{
char ch = 'A';
char b = tolower (ch);
printf ("upper case letter %c is converted to lower case letter %c", ch, b);
return 0;
}
```

Output on the screen:

upper case letter A is converted to lower case letter a

If you want to enter the character through the keyboard, then the above program should take the form:

```
#include<stdio.h>
int main ()
{
char ch;
printf ("Enter any character:");
scanf ("%c",      &   ch);
char b = tolower (ch);
printf ("upper case letter %c is converted to lower case letter %c", ch, b);
return 0;
}
```

Output on the screen:

Enter any character:

If you enter the character C

upper case letter C is converted to lower case letter c will be outputted on the screen.

Program 4.6

C program to convert the lower case letter to upper case letter

```
#include<stdio.h>
int main ()
{
char ch = 'a';
char b = toupper (ch);
printf ("lower case letter %c is converted to upper case letter %c", ch, b);
return 0;
}
```

Output on the screen:

```
lower case letter a is converted to upper case letter A
```

If you want to enter the character through the keyboard, then the above program should take the form:

```
#include<stdio.h>
int main ()
{
char ch;
printf ("Enter any character:");
scanf ("%c",      &   ch);
char b = toupper (ch);
printf ("lower case letter %c is converted to upper case letter %c", ch, b);
return 0;
}
```

Output on the screen:

```
Enter any character:
```

If you enter the character h

lower case letter h is converted to upper case letter H will be outputted on the screen.

Program 4.7

C program to test whether the entered character is upper case letter or not

```
#include<stdio.h>
int main ()
{
char ch = 'a';
if (isupper (ch))
printf ("you have entered the upper case letter");
else
printf ("you have entered the lower case letter");
return 0;
}
```

Output on the screen:

```
you have entered the lower case letter
```

If the statement:

char ch = 'a'; is replaced by the statement:

char ch = 'A';

Then the output on the screen is:

```
you have entered the upper case letter
```

Program 4.8

C program to test whether the entered character is lower case letter or not

```
#include<stdio.h>
int main ()
{
char ch = 'a';
if (islower (ch))
printf ("you have entered the lower case letter");
else
printf ("you have entered the upper case letter");
return 0;
}
```

Output on the screen:

```
you have entered the lower case letter
```

Program 4.9

C program to print the value of tan inverse x (i.e., the value of tan-1x)

```
#include<stdio.h>
#include<math.h>
int main ()
{
int x = 20;
printf ("the value of tan inverse x = %f", atan (x));
return 0;
}
```

Output on the screen:

```
the value of tan inverse x = 1.520838
```

Program 5.0

C program to print the value of tan inverse x/y (i.e., the value of tan-1x/y)

```
#include<stdio.h>
#include<math.h>
int main ()
{
int x,y;
x = 20;
y =20;
printf ("the value of tan inverse x/y = %f", atan2(x,y));
return 0;
}
```

Output on the screen:

```
the value of tan inverse x/y = 0.785398
```

Program 5.1

C program to print the value of fmod (x, y)

```
#include<stdio.h>
#include<math.h>
int main ()
{
float x = 20.500000;
float y =20.799999;
printf ("the remainder of %f divided by %f is %f", x, y, fmod (x,y));
return 0;
}
```

Output on the screen:

```
the remainder of 20.500000 divided by 20.799999 is 20.500000
```

Program 5.2

C program to print the value of ~x

```
#include<stdio.h>
int main ()
{
int x, y;
x = 205;
y=~x;
printf ("the value of y is:%d", y);
return 0;
}
```

Output on the screen:

```
the value of y is:-206
```

If the statement:

y=~x; is replaced by the statement:

y= -(~x);

Then the output on the screen is:

```
the value of y is: 206
```

Program 5.3

C program to print the ASCII (American Standard Code for Information Interchange) value of the entered character

```
#include<stdio.h>
int main ()
{
char ch ='A';
printf ("the ASCII value of ch is: %d", ch);
return 0;
}
```

Output on the screen:

```
the ASCII value of ch is: 65
```

If the statement:

printf ("the ASCII value of ch is: %d", ch);

is replaced by the statement:

printf ("the ASCII value of ch is: %c", ch);

Then the output on the screen is:

```
the ASCII value of ch is: A
```

What will be the output of the following programs:

i)

```
#include<stdio.h>
int main ()
{
int i;
int num [5] ={16,18,19,20,21};
for (i=0;i<5;i++)
printf ("\n Element = %d", num [i] +1);
return 0;
}
```

Answer:

```
Element = 17
Element = 19
Element = 20
Element = 21
Element = 22
```

ii)

```
#include<stdio.h>
int main ()
{
int i = 54;
int y = i<<1;
printf ("The value of y = %d", y);
return 0;
}
```

Answer:

```
The value of y = 108
```

If the statement:

i<<1 is replaced by the statement: i<<2

Then the output on the screen is:

```
The value of y = 216
```

Note:

i<<1 implies 54 * 2 = 108

i<<2 implies 54 * 4 = 216

i<<3 implies 54 * 6 = 324

i<<4 implies 54 * 8 = 432

iii)

```
#include<stdio.h>
int main ()
{
int i = 54;
int y = i>>1;
printf ("The value of y = %d", y);
return 0;
}
```

Answer:

```
The value of y = 27
```

If the statement:

i>>1 is replaced by the statement: i>>2

Then the output on the screen is:

```
The value of y = 13
```

Note:

i>>1 implies 54 / 2 = 27

i>>2 implies 54 / 4 = 13

i>>3 implies 54 / 6 = 9

i>>4 implies 54 / 8 = 6

<< implies: left shift operator

>> implies: right shift operator

Program 5.4

C program to print the length of the entered character (i.e., to print the length of the string)

```
#include<stdio.h>
#include<string.h>
int main ()
{
char ch [4];
printf ("Enter any word: ");
scanf ("%c",    &  ch);
printf ("The length of the string = %d", strlen (ch));
return 0;
}
```

Output on the screen:

```
Enter any word:
```

If you enter the word dog

```
The length of the string = 3
```

will be displayed on the console screen because there are three letters in the word dog.

Suppose if you enter the word tech

The length of the string = 4

will be displayed on the console screen because there are four letters in the word tech.

Program 5.5

C program to print the factorial of the entered number

```
#include<stdio.h>
int main ()
{
int i, n, fact=1;
printf ("Enter any number:");
scanf ("%d",    &   n);
for (i=1; i<=n; i++)
fact = fact *i;
printf ("\n Entered number is: %d", n);
printf ("\n The factorial of the entered number %d is: %d", n, fact);
return 0;
}
```

Output on the screen:

```
Enter any number:
```

If you enter the number 2

```
Entered number is: 2
The factorial of the entered number 2 is: 2
```

will be displayed on the screen.

Suppose if you enter the number 4

```
Entered number is: 4
The factorial of the entered number 4 is: 24
```

will be displayed on the screen.

What will be the output of the following program:

```
#include <stdio.h>
int main ()
{
printf ("\nLinux \' linux ");
printf ("\nLinux \? linux ");
return0;
}
```

Answer:

```
Linux ' linux
Linux? linux
```

```
#include<stdio.h>
#include<stdlib.h>
int main () {
printf ("linux\n");
exit (0);
printf ("php\n");
return 0;
}
```

Answer:

```
linux
```

Note: exit (0) is useful for terminating a program upon having discovered some error which prevents the program from continuing to execute normally. The header file for exit (0); is stdlib.h.

C++ Programming

An Object-oriented (Programming methodology that views a computer program as a combination of variables, functions, and data structures called objects) high level language (which uses alphabets, digits, punctuations and some special symbols and cannot be executed directly without being converted into machine level language (the language which uses only 0 and 1))

developed by a Danish computer scientist

Bjarne Stroustrup (in 1979 at AT & T Bell laboratories, USA) as an extension of the C language

initially named C with classes which later named C ++ in 1983.

As a successor of C language, C++ has been certified as a 99.9 percent pure standard and possesses exceptional performance, efficiency and flexibility of use compared to C language.

Advantage: Has the power and extensibility to write large-scale programs and runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX.

Uses: Used in the development of Apple Macintosh, PC running Windows, operating systems and Adobe Systems (like Photoshop, Acrobat etc).

C++ fully supports most important features of object-oriented programming including the four pillars of object-oriented development:

Encapsulation

Data hiding

Inheritance

Polymorphism

Inheritance

The ability of a class (sub class) to derive properties and characteristics from another class (super class) is called Inheritance. Inheritance is one of the most important feature of Object Oriented Programming.

The capability of a class to derive properties and characteristics from another class is calledInheritance. Inheritance is one of the most important feature of Object Oriented Programming.

Sub Class:The class that inherits properties from another class is called Sub class or Derived Class.

Super Class:The class whose properties are inherited by sub class is called Base Class or Super class.

Why and when to use inheritance?

Consider a group of vehicles. You need to create classes for Bus, Car and Truck. The methods fuelAmount (), capacity (), applyBrakes () will be same for all of the three classes. If we create these classes avoiding inheritance then we have to write all of these functions in each of the three classes as shown in below figure:

Process of C++ program execution: A C++ program:

```
#include<iostream>
int main ()
{
std::cout<<"Hello, crazy world!";
return 0;
}
```

is written using Text Editor, such as [Notepad++, Notepad] and saved with [.ccp] Extension.

File Saved with [.ccp] extension is called Source Program or Source Code.

C++ Source code with [.ccp] Extension is sent to preprocessor first.

The preprocessor generates an expanded source code:

```
//
----------------------------------------------------------------
----------------------------------------------------------------
The contents of <iostream>would be pasted at the location of #include<iostream>
int main ()
{
std::cout<<"Hello, crazy world!";
return 0;
}
----------------------------------------------------------
----------------------------------------------------------------
--- //
```

Expanded source code is given as input to compiler where the expanded source program is compiled (i.e., the program is entirely read and translated to instructions the computer can understand i.e., machine understandable / readable language i.e., to machine code sequence of 0s and 1s).

If the C++ compiler finds any error during compilation, it provides information about the error to the programmer.

The programmer has to review code and re-edit the program. After re-editing program, Compiler again check for any error.

If program is error-free then it is sent to assembler (where the code is assembled and converted into object code. Now a simple.obj file is generated).

The object code is sent to linker (where the object code is linked with included header files (such as iostream) and appropriate libraries).

Then it is converted into executable code. A simple.exe file is generated.

The executable code is sent to loader (where the executable code is loaded into memory and then it is executed).

After execution, output

```
Hello,world!
```

is displayed on the console screen.

Like C

C++ is case sensitive language: only lower case letters (or small letters) must be used.

Capital letters (or upper case letters) must be avoided to prevent the display of error on the screen

(For example: If the statement STD::COUT<<"Hello, crazy world!"; is written instead of

std::cout<<"Hello, crazy world!";

or INT MAIN () is written instead of int main (), compilation Error will be displayed on the console screen).

Parentheses () indicate a function and the word main indicate the name of the function.

main () implies: main function

And if we forget to end each statement within the body of the main function with a semicolon (;), then the compilation Error will be displayed on the screen.

There should be no space between main and the parentheses ()

i.e., int main ()

and there should be no space inside the parentheses ()

i.e., int main ( )

to prevent the display of compilation error on the screen.

As we know C++ is Platform dependent language. So the Operating system needs to know when the program execution ends.

So when there is value returns from the main function

the Operating System get to know that the program execution is over.

int main () implies: main () should return integer value.

If the main function returns 0 to the operating system, then the

program has completed execution successfully.

If the main function returns 1 to the operating system, then the

program has not completed execution successfully.

The statement #include<iostream> tells the compiler to include the text from the file iostream (which is already present in the operating system) before it translates or compiles the program into a sequence of 0s and 1s.

#include <iostream> is to C++ what #include<iostream> is to C (note one thing: there is no.h extension to the name iostream. The reason is that <iostream> is one of the modern style headers)

iostream means input output screen (i input, o output, stream screen) and iostream comprises input output functions like cout, cin etc. -- note: cin is a input function (cin means console input) and cout is a output function (cout means console output) and it is included into the C ++ program by writing the statement #include <iostream>). The statement #include tell the compiler to include the contents of the file iostream before compilation. If a program is written without the statement #include<iostream>, then the C++ compiler cant compile and a compilation error will be displayed on the screen (because C++ compiler fails to recognize the functions such as cin and cout).

int main () The program begins its execution with the function main () -- which is called the user defined function (because this function is defined by the user) - the main function -- the entry point of the program execution i.e., the point from where the execution of C++ program begins and the point at which the operating system passes control of the computer over to that program.

int main () {

} implies body of the main function within which the sequence of instructions in the form of statements

i.e., the program is written and executed. The left curly brace

{

implies: the beginning of the main function

and the right curly brace

}

implies: the end of the main function.

return 0; implies the exit status of execution of the program i.e., at this point,

main function returns back the control of the computer to the operating system since

the execution is terminated at this point and once a return statement

i.e., return 0; is executed, no further instructions within the main function are executed

For example:

```
#include<iostream>
int main ()
{
std::cout<<"Hello, crazy world!";
return 0;
std::cout<<"Hello, crazy world!";
}
```

Or

```
#include<iostream>
using namespace std;
int main ()
{
cout<<"Hello, crazy world!";
return 0;
cout<<"Hello, crazy world!";
}
```

Output on the screen:

```
Hello,world!
```

; implies semicolon --> A program is a well-defined set of instructions and each well-defined instruction (in the form of a statement)

is ended by a semicolon (which is C++ language punctuation --

like a period in English i.e., in an English paragraph each sentence is ended by a full stop which tells that one sentence ends and another begins,

semicolon implies that one instruction (or statement) ends and another begins).

cout implies the output function of the C++ language which makes provision to print the output:

Hello, crazy world!

on the console screen.

In the statement:

std::cout

std standard

:: scope resolution operator

cout console output

std::cout basically means: look in standard library and get cout function. The text

Hello, crazy world! should be enclosed by the double quotation marks ("") and if the statement:

using std::cout;

is added below the statement:

#include<iostream>

then the program takes the form:

```
#include<iostream>
using std::cout;
int main ()
{
cout<<"Hello, crazy world!";
return 0;
}
```

i.e., no need to include std:: in the statement:

std::cout<<"Hello, crazy world!";

Note: The symbol << implies: output the text:

Hello, crazy world!

on the console screen using the cout function.

Program 1.1

C++ program to print the word "hello Bill Gates" on screen

```
#include<iostream>
using std::cout;
int main ()
{
cout<<"hello Bill Gates";
return 0;
}
```

The output on the screen:

```
hello Bill Gates
```

Program 1.2

C++ program to print

```
*
*****
*****
*****
*****
```

on screen

```
#include<iostream>
using std::cout;
int main ()
{
cout<<"\n        *      ";
cout<<"\n ***** ";
cout<<"\n ***** ";
cout<<"\n ***** ";
cout<<"\n ***** ";
return 0;
}
```

The output on the screen:

```
*
*****
*****
*****
*****
```

If new line \n is not included in the above program then the output on the screen is:

```
*********************
```

Note:
endl
can be used instead of \n:

```
#include<iostream>
using std::cout;
int main ()
{
cout<<"   *      "<< endl;
cout<<"   ***** "<< endl;
cout<<"   ***** "<< endl;
cout<<"   ***** "<< endl;
cout<<"   ***** "<< endl;
return 0;
}
```

The output on the screen:
// The error:
endl was not declared in this scope //
will be displayed on the screen.
If the above program is rewritten:

```
#include<iostream>
using std::cout;
using std::endl;
int main ()
{
cout<<"   *      "<< endl;
cout<<"   ***** "<< endl;
cout<<"   ***** "<< endl;
cout<<"   ***** "<< endl;
cout<<"   ***** "<< endl;
return 0;
}
```

The output on the screen:

```
*
*****
*****
*****
*****
```

The single statement:
using namespace std;
can be used instead of the statements:
using std::cout;
using std::endl;
i.e.,

```
#include<iostream>
using namespace std;
int main ()
{
cout<<"   *      "<< endl;
cout<<"   ***** "<< endl;
cout<<"   ***** "<< endl;
cout<<"   ***** "<< endl;
cout<<"   ***** "<< endl;
return 0;
}
```

The output on the screen:

```
*
*****
*****
*****
*****
```

//--

cout <<"Hello world."<< endl; // Here it is necessary to put 'using namespace std' on the top of code.

std::cout <<"Hello world."<< std::endl; // Here there is no need to put 'using namespace std' on the top of code.

--//

Write a program to print the following outputs:

(a)

```
*
****
*******
****
*
```

(b)

```
***************
* *
* Hello World! *
* *
***************
```

(c)

Braces come in pairs!
Comments come in pairs!
All statements end with a semicolon!
Spaces are optional!
Must have a main function!
C++ is done mostly in lowercase. It's a

case-sensitive language

Answers:

(a)

```
#include<iostream>
using namespace std;
int main ()
{
cout<<"\n        *      ";
cout<<"\n        **** ";
cout<<"\n        ******* ";
cout<<"\n        ****       ";
cout<<"\n        *      ";
return 0;
}
```

(b)

```
#include<iostream>
using namespace std;
int main ()
{
cout<<"\n        ****************          ";
cout<<"\n        * *    ";
cout<<"\n        * Hello World! *       ";
cout<<"\n        * *    ";
cout<<"\n        ****************          ";
return 0;
}
```

(c)

```
#include<iostream>
using namespace std;
int main ()
{
cout<<"\n Braces come in pairs!";
cout<<"\n Comments come in pairs!";
cout<<"\n All statements end with a semicolon!";
cout<<"\n Spaces are optional!";
cout<<"\n Must have a main function!";
cout<<"\n C++ is done mostly in lowercase. It's a
case-sensitive language";
return 0;
}
```

Program 1.3

C++ program to find the area of a circle

```
#include<iostream>
using namespace std;
main ()
{
int r, area;
r = 2;
area = 4 * 3.14 * r * r;
cout<<"The area of the circle = "<< area;
return 0;
}
```

The output on the screen:

```
The area of the circle = 50
```

int means the data type is integer.

Note: An integer is a whole number -- no fractions, decimal parts, or funny stuff.

The statement

int r, area;

imply that we are creating the integer variables r, area.

Equal sign (" = ") implies storage operator.

The statements

r = 2;

area = 4 * 3.14 * r * r;

imply that we are storing the values to the created variables (i.e., we are storing the value 2 for r and 4 * 3.14 * r * r = 4 * 3.14 * 2 * 2 = 50 for area).

Comma in the statement

int r, area;

imply variable separator.

The statement

cout<<"The area of the circle = "<< area;

make provision to print the output:

The area of the circle = 50

on the screen.

The area of the circle is 50. 24 (for r = 2) but The area of the circle = 50 is displayed on the screen because data type int is used instead of float.

If the statement:

float r, area; is used instead of int r, area;

i.e.,

```
#include<iostream>
int main ()
{
float r, area;
r = 2;
area = 4 * 3.14 * r * r;
cout<<"The area of the circle = "<< area;
return 0;
}
```

Then the output on the screen:

```
The area of the circle = 50.24
```

float means the data type is float.

The statement

float r, area;

imply that we are creating the floating variables r, area.

(floating point variable means fractional variable or decimal number (for example: 1.5, 2.5, 3.5, 4.7 etc.) whereas integer means non-fractional variable or whole number (for example: 1, 2, 3, 4 etc.))

data type float is used instead of int because if the data type int is used instead of float then the result will not be clearly outputted i.e., instead of 50.24 the computer displays only 50.

If you want to supply the value for r through the key board, then the statement

float r = 2;

should be replaced by the statements

cout<<"Enter any number:";

```
cin>>r;
```

i.e.,

```
#include<iostream>
using namespace std;
int main ()
{
float r, area;
cout<<"Enter any number:";
cin>>r;
area = 4 * 3.14 * r * r;
cout<<"The area of the circle = "<< area;
return 0;
}
```

The output on the screen:

```
Enter any number:
```

If you the number 2

The area of the circle = 50.24 will be outputted on the screen.

As told earlier: cout is an output function and cin is an input function.

The statement:

```
cout<<"Enter any number:";
```

make provision to print the text

Enter any number:

on the screen.

cin>> r; is to C++ what scanf ("%d", & r); is to C

If you write the statement:

```
area = 4 * 3.14 * r ^ 2;
```

instead of

```
area = 4 * 3.14 * r * r;
```

Then compilation error will be displayed on the console screen because like in C Language there is no operator for performing exponentiation operation -- so the statement

area = 4 * 3.14 * r ^ 2; is invalid.

Note:

cout and cin are not part of C++ language but they are part of iostream file

Hence the statement #include<iostream> should be included in the C++ program otherwise cout and cin will not work and the compilation error will be displayed on the console screen.

Note:

Right shift operator >> denote stream extraction operator (extract data entered through the keyboard)

Left shift operator << denote stream insertion operator (insert data into an output screen)

<< and >> are termed overloaded operators and the file iostream defines these operators.

Note: As told earlier: when you enter an integer for x through the keyboard, this integer will be stored in the computer memory.

If you yearn to know the storage size of the integer in computer memory

(i.e., space occupied by the entered integer in the computer memory), you need to appeal to the following program:

```
#include<iostream>
using namespace std;
int main ()
{
int x;
x=10;
cout<<"size of r = "<< sizeof (r);
return 0;
}
```

The output on the screen:

```
size of x = 4
```

i.e., integer entered for r i.e., 10 has occupied a space of 4 bytes in the computer memory.

Write a program to print the circumference of the circle (given r = 2.5)

Answer:

```
#include<iostream>
using namespace std;
int main ()
{
float r, area;
r = 2.5;
circumference = 3.14 * r * r;
cout<<"The circumference of the circle = "<< circumference;
return 0;
}
```

Write a program to print the area of the rectangle (given l = 2.5 and b = 3)

Answer:

```
#include<iostream>
using namespace std;
int main ()
{
float l, b, area;
l = 2.5;
b = 3;
area = 1*b;
cout<<"The area of the rectangle = "<< area;
return 0;
}
```

Format Specifiers in C++

Data Type	Format Specifier
int	%d
float	%f or %e
char	%c
double	%lf or %le
long int	%ld

Program 4.6

C++ program to find the sum of two numbers

```
#include<iostream>
using namespace std;
```

```
int main ()
{
int a, b, sum;
a=1;
b=2;
sum = a + b;
cout<<"the sum of a and b = "<< sum;
return 0;
}
```

The output on the screen:

```
the sum of a and b = 3
```

If you assign the floating point values 1.5 & 2.6 for a & b, then the statement:

int a, b, sum; should be replaced by the statement float a, b, sum;

i.e.,

```
#include<iostream>
using namespace std;
int main ()
{
float a, b, sum;
a=1.5;
b=2.6;
sum = a + b;
cout<<"the sum of a and b = "<< sum;
return 0;
}
```

The output on the screen:

```
the sum of a and b = 4.1
```

The statement:

cout<<"the sum of a and b = "<< sum;

make provision to print the output:

the sum of a and b = 4.1

on the console screen. And if the statement:

cout<<"the sum of a and b = "<< sum;

is omitted from the C ++ program, then the program will be successfully executed but there will be no display of the output on the console screen.

If you want to supply the values for a and b through the key board, then the statements:

a=1.5;
b=2.6;

should be replaced by the statements:

cout<<"Enter any two numbers:";
cin>>a;
cin>>b;

i.e.,

```
#include<iostream>
using namespace std;
int main ()
{
float a, b, sum;
cout<<"Enter any two numbers:";
cin>>a;
cin>>b;
sum = a+ b;
cout<<"the sum of a and b = "<< sum;
return 0;
}
```

The output on the screen:

```
Enter any two numbers:
```

If you enter two numbers 2.9 & 3.6

the sum of a and b = 6.5

will be outputted on the screen.

The statement:

cout<<"Enter any two numbers:";

make provision to print

Enter any two numbers:

on the screen and the statements:

cin>>a;
cin>>b;

make provision to read the two numbers 2.9 and 3.6 entered through the keyboard and store them in the computer memory.

If the statements:

cout<<"Enter any two numbers:";
cin>>a;
cin>>b;

are replaced by the statements:

cout<<"Enter any number:";
cin>>a;
cout<<"Enter any number:";
cin>>b;

Then the output on the screen is:

Enter any number:

If you enter the number 2.9

Enter any number:

If you enter the number 3.6

the sum of a and b = 6.5

will be outputted on the screen.

If the statement:

cout<<"the sum of a and b = "<< sum;

is replaced by the statement:

cout<< a <<" + "<< b <<" = "<< sum;

Then the output:

2.9 + 3.6 = 6.5

will be displayed on the console screen.

What will be the output of the following program:

```
#include<iostream>
using namespace std;
int a = 5;
int main ()
{
int a =2;
cout<< a;
return 0;
}
```

Answer: 2

Note:

2 is a local variable (variable declared within the body of the main function)

The statement:

```
int a = 2;
```

imply: local variable declaration.

5 is a global variable (variable declared outside the body of the main function)

The statement:

```
int a = 5;
```

imply: global variable declaration.

If the statement:

```
cout<< a;
```

is replaced by the statement:

cout<<:: a; (where:: denote scope resolution operator)

i.e.,

```
#include<iostream>
using namespace std;
int a = 5;
int main ()
{
int a =2;
cout<<::a;
return 0;
}
```

Then the output on the screen is:

```
5
```

i.e., global variable will be outputted on the screen.

If the same program is written in C language

i.e.,

```
#include<stdio.h>
int a = 5;
int main ()
{
int a =2;
print ("%d",::a);
return 0;
}
```

Then the compilation error will be outputted on the screen because

scope resolution operator is not defined in the C language (i.e., C does not hold scope resolution operator).

Whether the following program will be successfully outputted or not:

```
#include<iostream>
using namespace std;
int main ()
{
int a, b, c;
a=3;
b=2;
c= a+b;
cout<<" sum of two numbers = 6"<< c;
return 0;
}
```

Answer:

Yes, the output on the screen is:

```
sum of two numbers = 65
```

Program 4.7

C ++ program to convert the temperature in Celsius to Fahrenheit

```
#include<iostream>
using namespace std;
int main ()
{
float C, F;
C=38.5;
F = 9*C/5 +32;
cout<<"temperature in Fahrenheit= "<< F;
return 0;
}
```

The output on the screen:

```
temperature in Fahrenheit = 101.3
```

As said Earlier:

If is used instead of *

and F = 9C/5 +32 is used of F = 9*C/5 +32, the compilation error will be displayed on the screen.

If you want to supply a value 16 digits after decimal point i.e., 36.5555555555555555 for C, then the statement:

```
double C, F;
```

should be used instead of the statement:

```
float C, F;
```

i.e.,

```
#include<iostream>
using namespace std;
int main ()
{
double C, F;
C=38.5555555555555555;
F = 9*C/5 +32;
cout<<"temperature in Fahrenheit= "<< F;
return 0;
}
```

If you want to supply the value for C through the key board, then the statement:

```
C=38.5;
```

should be replaced by the statements:

```
cout<<"Enter any number:";
cin>>C;
```

i.e.,

```
#include<iostream>
using namespace std;
int main ()
{
float C, F;
cout<<"Enter any number:";
cin>>C;
F = 9*C/5 +32;
cout<<"temperature in Fahrenheit= "<< F;
return 0;
}
```

The output on the screen:

```
Enter any number:
```

If you enter the number 23.6

```
temperature in Fahrenheit = 74.48
```

will be outputted on the screen.

Program 4.8

C++ program to find the product of two numbers

```
#include<iostream>
using namespace std;
int main ()
{
int a, b, product;
a=1;
b=2;
product = a * b;
cout<<"the product of a and b = "<< product;
return 0;
}
```

The output on the screen:

```
the product of a and b = 2
```

If you insert a value 2^3 for a and 3^2 for b, then as said earlier wrong result or compilation error will be flagged on the screen.

```
a=2^3;
b=3^2; ---> ERROR
a=2* 2*2
b=3*3; ---> Result will be outputted on the screen i.e.,
```

```
the product of a and b = 72
```

If you want to insert a 10 digit number for a and b i.e.,

```
a=1000000000
b=3000000000, then the statement:
int a, b, product;
```

should be replaced by the statement:

```
long int a, b, product;
```

i.e.,

```
#include<iostream>
using namespace std;
int main ()
{
long int a, b, product;
a=1;
b=2;
product = a * b;
cout<<"the product of a and b = "<< product;
return 0;
}
```

The output on the screen:

```
the product of a and b = 3000000000000000000
```

If you want to supply the integer values for a and b through the key board, then the statements:

```
a=1;
b=2; should be replaced by the statements:
cout<<"Enter any two numbers:";
cin >> a;
cin >> b;
```

i.e.,

```
#include<iostream>
using namespace std;
int main ()
{
int a, b, product;
cout<<"Enter any two numbers:";
cin>>a;
cin>>b;
product = a* b;
cout<<"the product of a and b = "<< product;
return 0;
}
```

The output on the screen:

```
Enter any two numbers:
```

If you enter two numbers 2 & 3

```
the product of a and b = 6
```

will be outputted on the screen.

If the statement:

```
cout<<the product of a and b = << product;
```

is written instead of the statement:

```
cout<<"the product of a and b = "<< product;
```

i.e., the statement the product of a and b = is not enclosed by the double quotation marks

Then the compilation error will be displayed on the console screen.

Program 4.9

C++ program to find the square of a number

```
#include<iostream>
using namespace std;
int main ()
{
int a, b;
a=2;
b = a * a;
cout<<"the square of a = "<< b;
return 0;
}
```

The output on the screen:

```
the square of a = 4
```

If you want to supply the integer value for a through the key board, then the statement:

```
a=2;
```

should be replaced by the statements:

```
cout<<"Enter any number:";
cin>>a;
```

i.e.,

```
#include<iostream>
using namespace std;
int main ()
{
```

```
int a, b;
cout<<"Enter any number:";
cin>>a;
b = a * a;
cout<<"the square of a = "<< b;
return 0;
}
```

The output on the screen:

```
Enter any number:
```

If you enter a number 3

```
the square of a = 9
```

will be outputted on the screen.

Note:

If the statement:

int main ();

is written instead of int main () then the error will be displayed on the screen

Write a program to print the cube of a number

Answer:

```
#include<iostream>
using namespace std;
int main ()
{
int a, b;
cout<<"Enter any number:";
cin>>a;
b = a * a*a;
cout<<"the cube of a = "<< b;
return 0;
}
```

Write a program to print the force applied to the mass m.

Answer:

```
#include<iostream>
using namespace std;
int main ()
{
int m, a, F;
cout<<"Enter the mass:";
cin>>m;
cout<<"Enter acceleration:";
cin>>a;
F = m * a;
cout<<"the force applied to the mass = "<< F;
return 0;
}
```

Program 5.0

C ++ program to find the greatest of two numbers using if - else statement

The syntax of if - else statement is:

```
if (this condition is true)
{
print this statement;
}
else
{
print this statement;
}
```

```
#include<iostream>
using namespace std;
int main ()
{
int a, b;
a = 2;
b = 3;
if (a>b)
{
cout<<"a is greater than b";
}
else
{
cout<<"b is greater than a";
}
return 0;
}
```

The output on the screen:

```
b is greater than a
```

Since the condition a>b within the parentheses is not true, the statement a is greater than b is not executed;

instead the execution skips and pass to print the statement b is greater than a.

In simpler words,

(a>b) is the condition (i.e., logical expression that results in true or false) and

if the condition (a>b) is true, then the statement

```
{
cout<<"a is greater than b";
}
```

is executed to print the output:

```
a is greater than b
```

else the statement

```
{
cout<<"b is greater than a";
}
```

is executed to print the output:

```
b is greater than a
```

If you want to supply the integer values for a and b through the key board, then the statements:

a=2;

b=3; should be replaced by the statements

```
cout<<"Enter any number:";
scanf ("%d",     &   a;
cout<<"Enter any number:";
scanf ("%d",     &   b;
```

i.e., the program should be rewritten as:

```
#include<iostream>
using namespace std;
int main ()
{
```

```
int a, b;
cout<<"Enter any number:";
scanf ("%d",   &   a;
cout<<"Enter any number:";
scanf ("%d",   &   b;
if (a>b)
{
cout<<"a is greater than b";
}
else
{
cout<<"b is greater than a";
}
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter the number 6
Enter any number:
If you enter the number 3
a is greater than b
```

will be outputted on the screen.

Program 5.1

C++ program to find the greatest of three numbers using if else if else statement

The syntax of if - else if - else statement:

```
if (this condition is true)
{
print this statement;
}
else if (this condition is true)
{
print this statement;
}
else
{
print this statement;
}
```

```
#include<iostream>
using namespace std;
int main ()
{
int a, b, c;
cout<<"Enter any number:";
cin>>a;
cout<<"Enter any number:";
cin>>b;
cout<<"Enter any number:";
cin>>c;
if (a>b   &   &   a>c)
{
cout<< a<<" is greater than"<< b<<" and "<<c;
}
else if (b>a   &   &   b>c)
{
cout<< b<<" is greater than"<< a <<" and "<<c;
}
else
{
cout<< c<<" is greater than"<< b<<" and "<< a;
}
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter the number 2
Enter any number:
If you enter the number 3
Enter any number:
If you enter the number 4
```

4 is greater than 3 and 2 will be outputted on the screen.

double ampersand " & & " imply:

and

(a>b & & a>c)

(b>a & & b>c)

denote conditions.

i.e., the condition

(a>b & & a>c) imply:

a is greater than b and a is greater than c

and if this condition is true, then the statement:

```
{
cout<< a<<" is greater than"<< b<<" and "<<c;
}
```

is executed to print the output:

a is greater than b and c

and if the condition (a>b & & a>c) is not true

the statement

```
{
cout<< a<<" is greater than"<< b<<" and "<<c;
}
```

is not executed; instead the execution skips and pass to the condition (b>a & & b>c)

and if this condition is true, then the statement:

```
{
cout<< b<<" is greater than"<< a <<" and "<<c;
}
```

is executed to print the output:

b is greater than a and c

and if the condition (b>a & & b>c) is not true, then the statement:

```
{
cout<< b<<" is greater than"<< a <<" and "<<c;
}
```

is not executed; instead the execution skips and the statement:

```
{
cout<< c<<" is greater than"<< b<<" and "<< a;
}
```

is executed to print the output:

c is greater than b and a

What will be the output of the following program?

```
#include <iostream>
int main ()
{
int a, b;
a=2;
b=2;
if (a>b || a= = b)
cout<<"a is greater than or equal to b";
else
cout<<"b is greater than a";
return 0;
}
```

Answer:

a is greater than or equal to b

Note: symbol || denote OR i.e., a>b || a = = b denote a is greater than or a is equal to b.

Program 5.2

C ++ program to find the average of 10 numbers

```
#include<iostream>
using namespace std;
int main ()
{
int N1, N2, N3, N4, N5, N6, N7, N8, N9, N10, X;
cout<<"Enter any 10 numbers:";
cin>>N1;
cin>>N2;
cin>>N3;
cin>>N4;
cin>>N5;
cin>>N6;
cin>>N7;
cin>>N8;
cin>>N9;
cin>>N10;
X = (N1 + N2 + N3 + N4 + N5 + N6 + N7 + N8 + N9 + N10) /10;
cout<<"the average of 10 numbers = "<< X;
return 0;
}
```

The output on the screen:

Enter any 10 numbers:

If you enter ten numbers 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10

the average of 10 numbers = 5

will be outputted on the screen.

Note: The average of 10 numbers is 5.5, the output on the screen is 5 because int is used instead of float.

Like in C language, any mathematical expression should be written in C ++ equivalent expression to prevent the display of compilation error on the screen because C ++ language also does not accept the general mathematical expressions.

Note: C++ equivalent mathematical expression is same as C equivalent mathematical expression

For example:

Mathematical expression:	C equivalent expression:	C++ equivalent expression:
log10x + bx	log10 (x) + b * x	log10 (x) + b * x

Program 5.3

C ++ program to find the square root of a number

```
#include<iostream>
#include<cmath>
using namespace std;
int main ()
{
int a, b;
cout<<"Enter any number:";
cin>> a;
b = sqrt (a);
cout<<"the square root of a number = "<< b;
return 0;
}
```

The output on the screen:

Enter any number:

If you enter the number 16

the square root of a number = 4

will be outputted on the screen.

Note:

This program can also be written as:

```
#include<iostream>
#include<cmath>
using namespace std;
int main ()
{
cout<<"the square root of a number = "<< sqrt (4);
return 0;
}
```

Suppose if you enter the number 8,

the square root of a number = 2

will be outputted instead of

the square root of a number = 2.82

on the screen because int is used instead of float.

Note: Since b = sqrt (a) is written

the statement:

#include<cmath> must be included in the above program because cmath file defines the mathematical functions like sqrt ().

If the statement:

#include<cmath> is not included in the above program:

```
#include<iostream>
using namespace std;
int main ()
{
int a, b;
cout<<"Enter any number:";
cin>> a;
b = sqrt (a);
cout<<"the square root of a number = "<< b;
return 0;
}
```

Then the compilation error will be displayed on the console screen.

Note:

#include<math.h> is used in C

whereas #include<cmath> is used in C ++

Write a program to print the cube root of a number:

Answer:

```
#include<iostream>
#include<cmath>
using namespace std;
int main ()
{
cout<<"the cube root of a number = "<< cbrt (8);
return 0;
}
```

Program 5.4

C++ program to find the simple interest

```
#include<iostream>
using namespace std;
int main ()
{
int P,T, R, SI;
P = 1000;
T = 2;
R = 3;
SI = P*T*R/100;
cout<<"the simple interest = "<< SI;
return 0;
}
```

The output on the screen:

```
the simple interest = 60
```

Note:

If you write:

SI = PTR/100;

instead of:

SI = P*T*R/100;

Then the compilation error is displayed on the screen because (like C) C ++ language does not accept the general expressions.

If you want to supply the integer values for P, T and R through the key board, then the statements:

```
P = 1000;
T = 2;
R = 3;
```

should be replaced by the statements:

```
cout<<"Enter principal amount:";
cin>>P;
cout<<"Enter time:";
cin>>T;
cout<<"Enter rate of interest:";
cin>>R;
```

i.e., the above program should take the form:

```
#include<iostream>
using namespace std;
int main ()
{
int P,T, R, SI;
cout<<"Enter principal amount:";
cin>>P;
cout<<"Enter time:";
cin>>T;
cout<<"Enter rate of interest:";
cin>>R;
SI = P*T*R/100;
cout<<"the simple interest = "<<SI;
return 0;
}
```

The output on the screen:

Enter principal amount:

If you enter the principal amount 1000

Enter time:

If you enter the time 2

Enter rate of interest:

If you enter the rate of interest 3

the simple interest = 60

will be outputted on the screen.

Program 5.5

C++ program to find the senior citizen

```
#include<iostream>
using namespace std;
int main ()
{
int age;
age=20;
if (age > = 60)
{
cout<<"senior citizen";
}
if (age<60)
{
cout<<"not a senior citizen";
}
return 0;
}
```

The output on the screen:

```
not a senior citizen
```

(age > = 60) means: age greater than or equal to 60.

If you want to supply the value for age through the key board, then the statement:

```
age = 20;
```

should be replaced by the statements:

```
cout<<"Enter age:";
cin>>age;
```

i.e.,

```
#include<iostream>
using namespace std;
int main ()
{
int age;
cout<<"Enter age:";
cin>>age;
if (age>60)
{
cout<<"senior citizen";
}
if (age<60)
{
cout<<"not a senior citizen";
}
return 0;
}
```

The output on the screen:

```
Enter age:
```

If you enter the age 60

```
senior citizen
```

will be outputted on the screen.

Suppose if you enter the age 31

```
not a senior citizen
```

will be outputted on the screen

Program 5.6

C ++ program to get marks for 3 subjects and declare the result.

If the marks >= 35 in all the subjects the student passes else fails.

```
#include<iostream>
using namespace std;
int main ()
{
int M1, M2, M3;
M1 = 38;
M2= 45;
M3 = 67;
if (M1 >= 35 & & M2>= 35 & & M3>= 35)
{
cout<<"candidate is passed";
}
else
{
cout<<"candidate is failed";
}
return 0;
}
```

The output on the screen:

```
candidate is passed
```

>= imply: greater than or equal to and double ampersand imply: and

(M1>= 35 & & M2>= 35 & & M3>= 35) denote the condition and this condition imply M1 is greater than or equal to 35

and M2 is greater than or equal to 35 and M3 is greater than or equal to 35. And if this condition is TRUE, then the statement

```
{
cout<<"candidate is passed";
}
```

is executed to print the output:

```
candidate is passed
```

else the statement:

```
{
cout<<"candidate is failed";
}
```

is executed to print the output:

```
candidate is failed
```

If you want to supply the integer values for marks M1, M2 and M3 through the key board, then the statements:

```
M1 = 38;
M2= 45;
M3 = 67;
```

should be replaced by the statements:

```
cout<<"Enter any three marks:";
cin>> M1;
cin>> M2;
cin>> M3;
```

i.e.,

```
#include<iostream>
int main ()
{
int M1, M2, M3;
cout<<"Enter any three marks:";
cin>> M1;
cin>> M2;
cin>> M3;
if (M1 >= 35 & & M2>= 35 & & M3>= 35)
{
cout<<"candidate is passed";
}
else
{
cout<<"candidate is failed";
}
return 0;
}
```

The output on the screen:

```
Enter any three numbers:
```

If you enter three numbers 26, 28, 39

```
candidate is failed
```

will be outputted on the screen.

Program 5.7

C ++ program to find profit or loss

```
#include<iostream>
using namespace std;
int main ()
{
int CP, SP, loss, profit;
cout<<"Enter cost price:";
cin >> CP;
cout<<"Enter selling price:";
cin>>SP;
if ( SP > CP )
{
cout<<"profit= "<< SP-CP;
}
else
{
cout<<"loss = "<< CP-SP;
}
return 0;
}
```

The output on the screen:

```
Enter cost price:
```

If you enter the cost price 25

```
Enter selling price:
```

If you enter the selling price 26

```
profit = 1
```

will be outputted on the screen.

If the condition (SP>CP) is true, then the statement:

```
{
cout<<"profit= "<< SP-CP;
}
```

is executed to print the output:

profit = SP-CP (in this case profit = 26-25 =1)

else the statement:

```
{
cout<<"loss = "<< CP-SP;
}
```

is executed to print the output:

loss = CP-SP

Program 5.8

C++ program to convert inches into centimeter

```
#include<iostream>
using namespace std;
int main ()
{
float I, C;
I=3.5;
C = 2.54*I;
cout<<"length in centimeters = "<< C;
return 0;
}
```

The output on the screen:

```
length in centimeters = 8.89
```

Note: float is used instead of int because I = 3.5 if int is used instead of float then the result will not be clearly outputted i.e., instead of 8.89 the computer displays only 8.

If you want to supply the value for I through the key board, then the above program should take the form:

```
#include<iostream>
using namespace std;
int main ()
{
float I, C;
cout<<"Enter the length in inches:";
cin >> I;
C = 2.54*I;
cout<<"length in centimeters= "<< C;
return 0;
}
```

The output on the screen:

```
Enter the length in inches:
```

If you enter the value for I i.e., 25.5

length in centimeters = 64.9 will be outputted on the screen.

Suppose

If you enter the value 25

The output on the screen:

```
length in centimeters = 63.5
```

Even if you enter the value 25 instead of 25.5, float should be used instead of int because if float is not used then

C = 63 will be outputted on the screen.

Program 5.9

C++ program to find the incremented and decremented values of two numbers

```
#include<iostream>
using namespace std;
int main ()
{
int a, b, c, d, e, f;
a = 10;
b=12;
c=a+1;
d=b+1;
e=a-1;
f=b-1;
cout<<"the incremented value of a = "<< c;
cout<<"the incremented value of b = "<< d;
cout<<"the decremented value of a = "<< e;
cout<<"the decremented value of b = "<< f;
return 0;
}
```

The output on the screen:

```
the incremented value of a = 11 the incremented value of b = 13 the decremented value of a = 9 the decremented value of b = 11
```

If the statements:

```
cout<<"the incremented value of a = "<< c;
cout<<"the incremented value of b = "<< d;
cout<<"the decremented value of a = "<< e;
cout<<"the decremented value of b = "<< f;
```

are replaced by the statements:

```
cout<<"\n the incremented value of a = "<< c;
cout<<"\n the incremented value of b = "<< d;
cout<<"\n the decremented value of a = "<< e;
cout<<"\n the decremented value of b = "<< f;
```

Then the output on the screen is:

```
the incremented value of a = 11
the incremented value of b = 13
the decremented value of a = 9
the decremented value of b = 11
```

If the statements:

```
cout<<"the incremented value of a = "<< c;
cout<<"the incremented value of b = "<< d;
cout<<"the decremented value of a = "<< e;
cout<<"the decremented value of b = "<< f;
```

are replaced by the statements:

```
cout<<"the incremented value of a = "<< c << endl;
cout<<"the incremented value of b = "<< d << endl;
cout<<"the decremented value of a = "<< e << endl;
cout<<"the decremented value of b = "<< f << endl;
```

Then the output on the screen:

```
the incremented value of a = 11
the incremented value of b = 13
the decremented value of a = 9
the decremented value of b = 11
```

If you want to supply the values for a and b through the key board,

then the above program should take the form:

```
#include<iostream>
using namespace std;
int main ()
{
int a, b, c, d, e, f;
cout<<"Enter any number:";
cin>> a;
cout<<"Enter any number:";
cin>> b;
c=a+1;
d=b+1;
e=a-1;
f=b-1;
cout<<"\n the incremented value of a = "<< c;
cout<<"\n the incremented value of b = "<< d;
cout<<"\n the decremented value of a = "<< e;
cout<<"\n the decremented value of b = "<< f;
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter the number 2
Enter any number:
If you enter the number 3
the incremented value of a = 3
the incremented value of b = 4
the decremented value of a = 1
the decremented value of b = 2
will be outputted on the screen.
```

Note:

b++ is same as b+1 and b-- is same as b-1

What will be the output of the following program:

```
#include<iostream>
using namespace std;
int main ()
{
float T1, T2, A;
cout<<"Enter any number:";
cin >>T1;
cout<<"Enter any number:";
cin >>T2;
A = (T1 + T2) / 2;
cout<<"the average temperature of the day = "<< A;
return 0;
}
```

Answer:

```
Enter any number:
If you enter the number:
2
Enter any number:
If you enter the number:
3
the average temperature of the day = 2.5
will be displayed on the console screen.
```

Program 6.0

The percentage marks are entered and the grades are allotted as follows:

percentage >= 60 First Class
percentage >=50 and per <= 60 Second Class
percentage >= 40 and per <= 50 Pass Class
percentage < 40 Fail

Write a C++ program for the above:

```
#include<iostream>
using namespace std;
main ()
{
int P;
cout<<"Enter the percentage:";
cin>>P;
if (P >= 60)
{
cout<<"first class";
}
if (P>=50  &  &  P <60)
{
cout<<"second class";
}
if (P>=40  &  &  P<=50 )
{
cout<<"pass class";
}
if (P<40)
{
cout<<"fail";
}
return 0;
}
```

The output on the screen:

```
Enter the percentage:
```

If you enter the percentage 35

```
fail
```

will be outputted on the screen.

Program 6.1

C++ program to calculate the discounted price and the total price after discount

Given:

If purchase value is greater than 1000, 10% discount

If purchase value is greater than 5000, 20% discount

If purchase value is greater than 10000, 30% discount

discounted price

```
#include<iostream>
using namespace std;
int main ()
{
double PV, dis;
cout<<"Enter purchased value:";
cin>>PV;
if (PV>1000)
{
cout<<"dis= "<< PV* 0.1;
}
else if (PV>5000)
{
cout<<"dis= "<< PV* 0.2;
}
else
{
cout<<"dis= "<< PV* 0.3;
}
return 0;
}
```

The output on the screen:

```
Enter purchased value:
```

If you enter the purchased value 6500

```
dis = 1300.000000
```

will be outputted on the screen.

(PV>1000), (PV>5000) denote the conditions and if the condition (PV>1000) is true i.e., purchased value is greater than 1000, then the statement

```
{
cout<<"dis= "<< PV* 0.1;
}
```

is executed to print the output:

dis= PV* 10% = PV* 10 /100 = PV* 0.1

and if the condition (PV>1000) is false and if the condition (PV>5000) is true i.e., purchased value is greater than 5000, then the statement

```
{
cout<<"dis= "<< PV* 0.2;
}
```

is executed to print the output:

dis= PV* 20% = PV* 20 /100 = PV* 0.2

and if the condition (PV>5000) is not true i.e., purchased value is less than 5000, then the statement

```
{
cout<<"dis= "<< PV* 0.3;
}
```

is executed to print the output:

dis= PV* 30% = PV* 30 /100 = PV* 0.3

total price

```
#include<iostream>
using namespace std;
int main ()
{
double PV, total;
cout<<"Enter purchased value:";
scanf ("%lf",  &  PV;
if (PV<1000)
{
cout<<"total= "<< PV - PV* 0.1;
}
else if (PV<5000)
{
cout<<"total = "<< PV- PV* 0.2;
}
else
{
cout<<"total= "<< PV- PV* 0.3;
}
return 0;
}
```

The output on the screen:

```
Enter purchased value:
```

If you enter the purchased value 650

```
total = 585.000000
```

will be outputted on the screen.

If the condition (PV>1000) is true i.e., purchased value is greater than 1000, then the statement

```
{
cout<<"total = %d", PV - PV* 0.1;
}
```

is executed to print the output:

total =PV- dis = PV- PV*10% = PV- PV* 10 /100 = PV - PV * 0.1

and if the condition (PV>1000) is false and if the condition (PV>5000) is true i.e., purchased value is greater than 5000, then the statement

```
{
cout<<"total = %d", PV - PV* 0.2;
}
```

is executed to print the output:

total =PV- dis = PV- PV*20% = PV- PV* 20 /100 = PV - PV * 0.2

and if the condition (PV> 5000) is not true i.e., purchased value is less than 5000, then the statement

```
{
cout<<"total = %d", PV - PV* 0.3;
}
```

is executed to print the output:

total =PV- dis = PV- PV*30% = PV- PV* 30 /100 = PV - PV * 0.3

Now, Combing both the programs (above), we can write:

```
#include<iostream>
using namespace std;
int main ()
{
double PV, dis, total;
cout<<"Enter purchased value:";
cin>>PV;
if (PV>1000)
{
cout<<"dis= "<< PV* 0.1;
cout<<"total= "<< PV - PV* 0.1;
}
else if (PV>5000)
{
cout<<"dis = "<< PV* 0.2;
cout<<"total= "<< PV - PV* 0.1;
}
else
{
cout<<"dis= "<< PV* 0.3;
cout<<"total= "<< PV - PV* 0.1;
}
return 0;
}
```

The output on the screen:

Enter purchased value:

If you enter the purchased value 850

dis = 85.000000

total = 765.000000

will be outputted on the screen.

Program 6.2

C++ program to print the first ten natural numbers using for loop statement

```
#include<iostream>
using namespace std;
int main ()
{
int i;
for (i=1; i<=10; i++)
cout<<"value of i = "<< i;
return 0;
}
```

The output on the screen is:

```
value of i = 1 value of i = 2 value of i = 3 value of i = 4 value of i= 5 value of i= 6 value of i = 7 value of i= 8 value of i= 9 value of i= 10
```

for (i=1; i<=10; i++) denote the for loop statement and the syntax of the for loop statement is:

for (initialization; condition; increment)

Here:

i=1 denote initialization (i.e., from where to start)

i<=10 denote the condition (i.e., stop when 10 is reached)

i++ implies increment (which tells the value of i to increase by 1 each time the loop is executed) and i++ is the same as i+1.

When for loop executes, the following occurs:

i = 1

Is the condition (i<=10) is true?

Yes because i=1

The statement cout<<"value of i = "<< i; is executed to print the output:

value of i = 1

Now, the value of i is:

i = 1+1 = 2

Is the condition (i<=10) is true?

Yes because i=2

The statement cout<<"value of i = "<< i; is executed to print the output:

value of i = 2

Now, the value of i is:

i = 2+1 = 3

Is the condition (i<=10) is true?

Yes because i=3

The statement cout<<"value of i = "<< i; is executed to print the output:

value of i = 3

Now, the value of i is:

i = 3+1 = 4

Is the condition (i<=10) is true?

Yes because i=4

The statement cout<<"value of i = "<< i; is executed to print the output:

value of i = 4

Now, the value of i is:

i = 4+1 = 5

Is the condition (i<=10) is true?

Yes because i=5

The statement cout<<"value of i = "<< i; is executed to print the output:

value of i = 5

Now, the value of i is:

i = 5+1 = 6

Is the condition (i<=10) is true?

Yes because i=6

The statement cout<<"value of i = "<< i; is executed to print the output:

value of i = 6

Now, the value of i is:

i = 6+1 = 7

Is the condition (i<=10) is true?

Yes because i=7

The statement cout<<"value of i = "<< i; is executed to print the output:

value of i = 7

Now, the value of i is:

i = 7+1 = 8

Is the condition (i<=10) is true?

Yes because i=8

The statement cout<<"value of i = "<< i; is executed to print the output:

value of i = 8

Now, the value of i is:

i = 8+1 = 9

Is the condition (i<=10) is true?

Yes because i=9

The statement cout<<"value of i = "<< i; is executed to print the output:

value of i = 9

Now, the value of i is:

i = 9+1 = 10

Is the condition (i<=10) is true?

Yes because i=10

The statement cout<<"value of i = "<< i; is executed to print the output:

value of i = 10

and stop because the condition i<=10 is achieved.

If the statement:

cout<<"value of i = "<< i;

is replaced by the statement:

cout<<"\n value of i = "<< i;

Then the output on the screen is:

```
value of i = 1
value of i = 2
value of i = 3
value of i = 4
value of i = 5
value of i = 6
value of i = 7
value of i = 8
value of i = 9
value of i = 10
```

If the

for loop statement:

for (i=2; i<=10; i++)

is written instead of the statement:

for (i=1; i<=10; i++), then the output on the screen is:

value of i = 2 value of i = 3 value of i= 4 value of i= 5 value of i= 6 value of i = 7 value of i= 8 value of i = 9 value of i= 10

If the for loop statement:

for (i=1; i<10; i++)

is written instead of the statement:

for (i=1; i<=10; i++), then the output on the screen is:

value of i = 1 value of i = 2 value of i= 3 value of i= 4 value of i= 5 value of i= 6 value of i = 7 value of i= 8 value of i = 9

(Note: the condition i<=10 tells to print till value of i =10 but the condition i<10 tells to print till value of i=9)

If the statement:

for (i=1; i=10; i++)

is written instead of the statement:

for (i=1; i<=10; i++), then the output on the screen is:

value of i = 10 value of i = 10 value of i = 10 value of i = 10 value of i= 10 value of i= 10 value of i = 10 value of i= 10 value of i = 10 value of i = 10 value of i = 10 value of i = 10 value of i = 10 value of i = 10 value of i = 10 (continues...).

Note:

If the statement:

cout<<"value of i = "<< i; is replaced by the statement:

cout<<"\n "<< i;

Then the output on the screen is:

```
1
2
3
4
5
6
7
8
9
10
```

What will be the output of the following program:

```
#include<iostream>
using namespace std;
```

```
int main ()
{
int i;
for (i =1; i<=5; i ++)
cout<<"\n Linux is not portable";
return 0;
}
```

Answer:

```
Linux is not portable
Linux is not portable
Linux is not portable
Linux is not portable
Linux is not portable
```

C++ program to print the first ten natural numbers using for while loop statement

The syntax of while loop statement is:

```
while (this is the condition)
{
execute this statement;
}
```

```
#include<iostream>
using namespace std;
int main ()
{
int i = 1;
while (i<=10)
{
cout<<"\n "<< i++;
}
return 0;
}
```

The output on the screen is:

```
1
2
3
4
5
6
7
8
9
10
```

(i<=10) is the condition and

The statement

cout<<"\n "<< i++;

is repeatedly executed as long as a given condition (i<=10) is true.

If the statement:

int i=1;

is replaced by the statement:

int i;

Then the compilation error will be displayed on the console screen because initialization is not defined i.e., from where to start is not declared.

If the statement:

int i = 1;

is replaced by the int i = 0;

Then the output on the screen is:

```
0
1
2
3
4
5
6
7
8
9
10
```

Similarly if the statement int i = 0; is replaced by the int i = 7;

Then the output on the screen is:

```
7
8
9
10
```

C++ program to print first 10 numbers using do while loop statement

The syntax of do while loop statement is:

```
do
{
execute this statement;
}
while (this is the condition;
```

```
#include<iostream>
using namespace std;
int main ()
{
int i =1;
do
{
cout<<" \n i= "<< i++;
} while (i<=10);
return 0;
}
```

The output on the screen is:

```
i=1
i=2
i=3
i=4
i=5
i=6
i=7
i=8
i=9
i=10
```

The statement:

cout<<" \ni= "<< i++;

is executed and then condition (i<=10) is checked. If condition (i<=10) is true then

The statement:

cout<<" \ni= "<< i++;

is executed again. This process repeats until the given condition (i<=10) becomes false.

Write a program to print

When in doubt use brute force

100 times using for loop statement.

Answer:

```
#include<iostream>
using namespace std;
int main ()
{
int i;
for (i=0; i<=99; i++)
cout<<"\n When in doubt use brute force";
return 0;
}
```

Program 6.3

C++ program to print the characters from A to Z using for loop, do while loop and while loop statement.

C ++ program to print the characters from A to Z using for loop statement:

```
#include<iostream>
using namespace std;
int main ()
{
char a;
for ( a='A'; a<='Z'; a++)
cout<<" \n"<< a;
return 0;
}
```

The output on the screen:

```
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
W
X
Y
Z
```

char means the data type is character.

The statement:

char a;

imply that we are creating the character a.

If the statement:

for (a=A; a<=Z; a++) is written instead of the statement for (a='A'; a<='Z'; a++)

i.e., A is used instead of 'A' and Z is used instead of 'Z', then the compilation error will be displayed on the screen.

C ++ program to print the characters from A to Z using while loop statement:

```
#include<iostream>
using namespace std;
int main ()
{
char a = 'A';
while (a<='Z')
{
cout<<" \n"<< a++;
}
return 0;
}
```

C ++ program to print the characters from A to Z using do while loop statement:

```
#include<iostream>
using namespace std;
int main ()
{
char a = 'A';
do
{
cout<<" \n"<< a++;
} while (a<='Z');
return 0;
}
```

Program 6.4

C++ program to print the given number is even or odd.

```
#include<iostream>
using namespace std;
int main ()
{
int a;
cout<<"Enter any number:";
cin>>a;
if (a%2 = = 0)
{
cout<<"the number is even";
}
else
{
cout<<"the number is odd";
}
return 0;
}
```

The output on the screen:

Enter any number:
If you enter the number 6
the number is even
will be outputted on the screen.

(a%2 = = 0) is the condition and this condition imply: a divided by 2 yields reminder = 0.

For example: if you enter the number 2
Then a = 2
Then 2 divided by 2 yields the remainder = 0
Then the statement

```
{
cout<<"the number is even";
}
```

is executed to print the output:
the number is even

(Note: (like in C) in C ++ language = = implies: equal to)

If you enter the number 3
Then a = 3
Then 3 divided by 2 yields the remainder = 1
Then the statement

```
{
cout<<"the number is odd";
}
```

is executed to print the output:
the number is odd

Program 6.5

C++ program to print the remainder of two numbers

```
#include<iostream>
using namespace std;
int main ()
{
int a, b, c;
cout<<"Enter any number:";
cin>>a;
cout<<"Enter any number:";
cin>>b;
c = a % b;
cout<<"the remainder of a and b = "<< c;
return 0;
}
```

The output on the screen:
Enter any number:
If you enter the number 3
Enter any number:
If you enter the number 2
the remainder of a and b = 1
will be outputted on the screen.

Since (a =3 and b =2). Therefore:

3 divided by 2 (i.e., a divided by b) yields the remainder equal to 1

If the statement:

cout<<"the remainder of a and b = "<< c; is replaced by the statement:

cout <<" the remainder of "<<a <<"and"<< b <<"= "<< c;

i.e.,

```
#include<iostream>
using namespace std;
int main ()
{
int a, b, c;
cout<<"Enter any number:";
cin>>a;
cout<<"Enter any number:";
cin>>b;
c = a % b;
cout <<" the remainder of "<<a <<"and"<< b
<<"= "<< c;
return 0;
}
```

The output on the screen:
Enter any number:
If you enter the number 3
Enter any number:
If you enter the number 2
the remainder of 3 and 2 = 1
will be outputted on the screen.

Program 6.6

C++ program to check equivalence of two numbers

```
#include<iostream>
using namespace std;
int main ()
{
int x, y;
cout<<"Enter any number:";
cin>>x;
cout<<"Enter any number:";
cin>>y;
if (x-y==0)
{
cout<<"the two numbers are equivalent";
}
else
{
cout<<"the number are not equivalent";
}
return 0;
}
```

The output on the screen:
Enter any number:
If you enter the number 2
Enter any number:
If you enter the number 2
the two numbers are equivalent
will be outputted on the screen.

Since 2-2 is equal to 0 (i.e., x-y = = 0). Therefore: the statement

```
{
cout<<"the two numbers are equivalent";
}
```

is executed to print the output:

two numbers are equivalent

If you enter the numbers 3 and 2

The output on the screen:

the two numbers are not equivalent

Since 3-2 is not equal to 0 (i.e., x-y!= 0). Therefore: the statement

```
{
cout<<"the two numbers are not equivalent";
}
```

is executed to print the output:

two numbers are not equivalent

(Note: (like in C) in C ++ language!= implies not equal to)

Program 6.7

C ++ program to print whether the given number is positive or negative

```
#include<iostream>
using namespace std;
int main ()
{
int a;
a = -35;
if (a>0)
{
cout<<"number is positive";
}
else
{
cout<<" number entered is negative";
}
return 0;
}
```

The output on the screen:

number entered is negative

Since a = -35. Therefore:

a is less than 0 i.e., a < 0 because any negative number is always less than zero.

The statement:

```
{
cout<<"number is negative";
}
```

is executed to print the output:

number entered is negative

Program 6.8

C++ program to print the sum of the first 10 numbers using for loop statement

```
#include<iostream>
using namespace std;
int main ()
{
int i, sum = 0;
for ( i=1; i<=10; i++)
sum = sum + i;
cout<<"sum of the first10 numbers = "<< sum;
return 0;
}
```

The output on the screen:

sum of the first 10 digits = 55

How the sum of the first 10 digits = 55 is outputted on the screen through the for Loop statement?

i=1 (sum = 0 because the sum is initialized to 0 in the statement int i, sum = 0;)

Is i<=10 true?

Yes, do this

sum = sum + i = 0 +1 =1

Now,

i=2 (sum = 1)

Is i<=10 true?

Yes, do this

sum = sum + i = 1 +2 =3

Now,

i=3 (sum = 3)

Is i<=10 true?

Yes, do this

sum = sum + i = 3 +3 = 6

Now,

i=4 (sum = 6)

Is i<=10 true?

Yes, do this

sum = sum + i = 6 + 4= 10

Now,

i=5 (sum = 10)

Is i<=10 true?

Yes, do this

sum = sum + i = 10 + 5= 15

Now,

i=6 (sum = 15)

Is i<=10 true?

Yes, do this

sum = sum + i = 15 + 6 = 21

Now,

i=7 (sum = 21)

Is i<=10 true?

Yes, do this

sum = sum + i = 21 + 7 = 28

Now,

i=8 (sum = 28)

Is i<=10 true?

Yes, do this

sum = sum + i = 28 + 8 = 36

Now,

i=9 (sum = 36)

Is i<=10 true?

Yes, do this

sum = sum + i = 36 + 9 = 45

Now,

i=10 (sum = 45)

Is i<=10 true?

Yes, do this

sum = sum + i = 45 + 10 = 55
stops because the condition i<=10 is achieved
The statement:
cout<<"sum of the first 10 digits = "<< sum;
is executed to print the output:
sum of the first 10 digits = 55

If the statement:
int i, sum = 0;
is replaced by int i, sum = 1;
Then the output on the screen is:
sum of the first10 digits = 56
What will be the output if the for loop statement for (i =1; i<=10; i++) is replaced by the statement for (i =2; i<10; i++)?
Answer: sum of 10 digits = 44
If the statement int i, sum, sum = 0; is written instead of int i, sum = 0;
Then the compilation error message will be displayed on the screen (stating that sum is twice declared).
If the for loop is ended with a semicolon i.e.,
for (i=1; i<=10; i++;
Then the compilation error will be displayed on the console screen.
//---
sum = sum + a; is the same as sum + = a;
sub = sub - a; is the same as sub - = a;
product = product* a; is the same as product * = a;
div = div / a; is the same as div /= a;
a = a% b; is the same as a % = b;
---//
Program 6.9
C++ program to print the average of the first 10 numbers using for loop statement

```
#include<iostream>
using namespace std;
int main ()
{
int i, avg, sum = 0;
for ( i=1; i<=10; i++)
sum = sum + i;
avg = sum/10;
cout<<"sum of the first 10 numbers = "<< sum;
cout<<"average of the first 10 numbers = "<< avg;
return 0;
}
```

The output on the screen:
sum of the first 10 numbers = 55
average of the first 10 numbers = 5
The average of the first10 numbers = 55/10 = 5.5 not 5. But the output on the screen is:
average of the first 10 numbers = 5
because int is used instead of float.
If the data type float is used i.e.,

```
#include<iostream>
using namespace std;
int main ()
{
float i, avg, sum = 0;
for ( i=1; i<=10; i++)
sum = sum + i;
avg = sum/10;
cout<<"sum of the first 10 numbers = "<< sum;
cout<<"average of the first 10 numbers = "<< avg;
return 0;
}
```

The output on the screen:
sum of the first 10 numbers = 55
average of the first 10 numbers = 5.5
Program 7.0
C++ program to print the product of the first 10 digits using for loop statement

```
#include<iostream>
using namespace std;
int main ()
{
int i, product = 1;
for ( i=1; i<=10; i++)
product = product * i;
cout<<"the product of the first 10 digits =%d", product;
return 0;
}
```

The output on the screen:
the product of the first 10 digits = 3628800
How the product of the first 10 digits = 3628800 is outputted on the screen through the for Loop statement?
i=1 (product = 1 because the product is initialized to 1 in the statement int i, product = 1;)
Is i<=10 true?
Yes, do this
product = product * i = 1 * 1 =1
Now,
i=2 (product = 1)
Is i<=10 true?
Yes, do this
product = product * i = 1 * 2 = 2
Now,
i=3 (product = 2)
Is i<=10 true?
Yes, do this
product = product * i = 2 * 3 = 6
Now,
i=4 (product = 6)
Is i<=10 true?

Yes, do this
product = product * i = 6 * 4 = 24
Now,
i=5 (product =24)
Is i<=10 true?
Yes, do this
product = product * i = 24 * 5 =120
Now,
i=6 (product =120)
Is i<=10 true?
Yes, do this
product = product * i = 120 * 6 = 720
Now,
i=7 (product =720)
Is i<=10 true?
Yes, do this
product = product * i = 720 * 7 = 5040
Now,
i=8 (product =5040)
Is i<=10 true?
Yes, do this
product = product * i = 5040 * 8 = 40320
Now,
i=9 (product = 40320)
Is i<=10 true?
Yes, do this
product = product * i = 40320 * 9 = 362880
Now,
i=10 (product = 362880)
Is i<=10 true?
Yes, do this
product = product * i = 362880 * 10 = 3628800
stops because the condition i<=10 is achieved.

The statement:

cout<<"the product of the first 10 digits = "<< product; is executed to display the output:

the product of the first 10 digits = 3628800

If the statement:

int i, product = 1; is replaced by int i, product = 0;

Then the output on the screen is:

the product of the first 10 digits = 0

If the statement:

for (i=1; i<=10; i++) is replaced by for (i=5; i<=8; i++)

Then the output on the screen is:

the product of the first 10 digits = 1680

Program 7.1

C++ Program to print the table of a number using the for loop statement

```
#include<iostream>
using namespace std;
int main ()
{
int n, i;
cout<<"Enter any number:";
cin>>n;
for ( i=1; i<=5; i++)
cout<< n <<" * "<< i <<" = "<< n*i;
return 0;
}
```

The output on the screen:

```
Enter any number:
```

If you enter the number 2 (i.e., n=2)

```
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
```

will be outputted on the screen.

How the execution takes its Way through the for Loop statement

Since you entered the number 2, therefore: n=2.

i=1
Is i<=5 true?
Yes, print this
2 * 1 = 2
using the statement cout<< n <<" * "<< i <<" = "<< n*i;
Now,
i=2
Is i<=5 true?
Yes, print this
2 * 2 = 4
using the statement cout<< n <<" * "<< i <<" = "<< n*i;
Now,
i=3
Is i<=5 true?
Yes, print this
2 * 3 = 6
using the statement cout<< n <<" * "<< i <<" = "<< n*i;
Now,
i=4
Is i<=5 true?
Yes, print this
2 * 4 = 8
using the statement cout<< n <<" * "<< i <<" = "<< n*i;
Now,
i=5
Is i<=5 true?
Yes, print this
2 * 5 = 10
using the statement cout<< n <<" * "<< i <<" = "<< n*i;
stop Now because the condition i<=5 is achieved.

If the symbol * is replaced by +

i.e.,

```
#include<iostream>
```

```
using namespace std;
int main ()
{
int n, a;
cout<<"Enter any number:";
cin>>n;
for ( i=1; i<=5; i++)
cout<< n <<" +"<< i <<" = "<< n + i <<endl;
return 0;
}
```

Then the output on the screen is:

```
Enter any number:
```

If you enter the number 2 (i.e., n=2)

```
2 + 1 = 3
2 + 2 = 4
2 + 3 = 5
2 + 4 = 6
2 + 5 = 7
```

will be outputted on the screen.

Program 7.2

C++ program:

If you enter a character M

Output must be: ch = M

```
#include<iostream>
using namespace std;
int main ()
{
char M;
cout<<"Enter any character:";
cin>>M;
cout<<"ch= "<< M;
return 0;
}
```

The output on the screen:

```
Enter any character:
```

If you enter the character M

```
ch = M
```

will be outputted on the screen.

Note:

If we replace the statement:

cin>>M; by the statement:

M = getchar ();

i.e.,

```
#include<iostream>
using namespace std;
int main ()
{
char M;
cout<<"Enter any character:";
M = getchar ();
cout<<"ch= "<< M;
return 0;
}
```

There will be no change in the output on the screen i.e., The output on the screen is:

```
Enter any character:
```

If you enter the character K

```
ch = K
```

will be outputted on the screen.

If we replace the statement:

cout<<"ch= "<< M; by the statement:

putchar (M); i.e.,

```
#include<iostream>
using namespace std;
int main ()
{
char M;
cout<<"Enter any character:";
cin>>M;
putchar (M);
return 0;
}
```

There will be no change in the output on the screen i.e., The output on the screen is:

```
Enter any character:
```

If you enter the character M

M will be outputted on the console screen.

If we replace the statement:

cin>>M; by the statement:

M = getchar ();

and the statement:

cout<<"ch= "<< M; by the statement:

putchar (M); i.e.,

```
#include<iostream>
using namespace std;
int main ()
{
char M;
cout<<"Enter any character:";
M = getchar ();
putchar (M);
return 0;
}
```

The output on the screen:

```
Enter any character:
```

If you enter the character S

S will be outputted on the screen.

Write a program to print the absolute value of a number

Answer:

```
#include<iostream>
#include<cmath>
using namespace std;
int main ()
{
int a, b;
a= - 2;
b= abs (a);
cout<<" absolute value of a = "<< b<< endl;
return 0;
}
```

The output on the screen:

absolute value of a = 2

Program 7.2

C ++ program to print the first 5 numbers starting from one together with their squares

```
#include<iostream>
using namespace std;
int main ()
{
int i;
for ( i=1; i<=5; i++)
cout<<"\n number = "<< i <<"its square = "<<
i*i;
return 0;
}
```

The output on the screen:

```
number=1 its square=1
number=2 its square=4
number=3 its square=9
number=4 its square=16
number=5 its square=25
```

How the execution takes its way through the for loop statement

i=1

Is i<=5 true?

Yes, print this

number=1 its square=1

using the statement cout<<"\n number = "<< i <<"its square = "<< i*i;

Now,

i=2

Is i<=5 true?

Yes, print this

number=2 its square=4

using the statement cout<<"\n number = "<< i <<"its square = "<< i*i;

Now,

i=3

Is i<=5 true?

Yes, print this

number=3 its square=9

using the statement cout<<"\n number = "<< i <<"its square = "<< i*i;

Now,

i=4

Is i<=5 true?

Yes, print this

number=4 its square=16

using the statement cout<<"\n number = "<< i <<"its square = "<< i*i;

Now,

i=5

Is i<=5 true?

Yes, print this

number=5 its square=25

using the statement cout<<"\n number = "<< i <<"its square = "<< i*i;

stop Now because the condition (i<=5) is achieved.

Note:

If the statement cout<<"\n number = "<< i <<"its square = "<< i*i; is replaced by the statement:

cout<<"\n number = "<< i <<"\t its square = "<< i*i;

Then the output on the screen is:

```
number=1        its square=1
number=2        its square=4
number=3        its square=9
number=4        its square=16
number=5        its square=25
```

tab /t is included because to leave space between

number=1and its square=1

Suppose cout<<"\n number = "<< i <<"\t its square = "<< i*i; is replaced by the statement

cout<<"\n number = "<< i <<"\n its square = "<< i*i;

Then the output on the screen is:

```
number=1
its square=1
number=2
its square=4
number=3
its square=9
number=4
its square=16
number=5
its square=25
```

Write a program to print the first 10 numbers starting from one together with their squares and cubes?

Answer:

```
#include<iostream>
using namespace std;
int main ()
{
int i;
for ( i=1; i<=10; i++)
cout<<"number = "<< i <<" its square = "<< i*i
<<" its cube = "<< i*i*i<< endl;
return 0;
}
```

Program 7.3

C++ program to print the sum of two numbers using pointers

If we create an integer variable x by declaring the statement:

int x;

within the body of the main function "main ()" -- this variable is stored in the computer memory i.e.,

this variable occupies a specific location in the space of computer memory.

And this integer variable x is assigned an address (i.e., & x) to locate its position in the computer memory

(like a house in the street is assigned an address to locate its position in the street).

Pointers are the variables that represent the address of x in the computer memory i.e., p = & x,

where & x imply the address of x in the computer memory and

p is the pointer variable (which is the variable that represent the address of x in the computer memory).

And further if you assign a value to the variable x by declaring the statement x=1; within the body of the

main functionthis value is stored in the address of x in the computer memory. "*" denote pointer operator and *p denote the pointer

(which represent the value stored in the address of x in the computer memory).

C ++ program to print the address of x and the value assigned to x

```
#include<iostream>
using namespace std;
int main ()
{
int x, *p;
cout<<"Enter any integer:";
cin>>x;
p = & x;
cout<<"The address of the variable x = "<< p;
cout<<"The value of the variable x = "<< *p;
return 0;
}
```

The output on the screen:

```
Enter any integer:
If you enter the integer 1
The address of the variable x = 0x7fffc60478a4
The value of the variable x = 1
```

will be outputted on the screen.

The value of the variable x = 1 because you have assigned the value 1 to the variable x by entering 1 through the keyboard.

If the statements:

```
cout<<"The address of the variable x = "<< p;
cout<<"The value of the variable x = "<< *p;
```

are replaced by the statement:

```
cout<<"The address of the variable x = "<< p <<"its value = "<< *p;
```

i.e.,

```
#include<iostream>
using namespace std;
int main ()
{
int x, *p;
cout<<"Enter any integer:";
cin >> x;
p = & x;
cout<<"The address of the variable x = "<< p <<"its value = "<< *p;
return 0;
}
```

Then the output on the screen is:

```
The address of the variable x = 0x7fff78508cc4 its value = 2
```

```
#include<iostream>
using namespace std;
int main ()
{
int x, y, *p, *q, sum;
cout<<"Enter any number:";
cin >> x;
cout<<"Enter any number:";
cin >> y;
p = & x;
q = & y;
sum = *p + *q;
cout<<"\n sum of entered numbers = "<< sum;
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter the number 4
Enter any number:
If you enter the number 3
sum of entered numbers = 7
```

will be outputted on the screen.

Since *p imply the value assigned to the variable x (i.e., 4) by entering 4 through the keyboard

and *q imply the value assigned to the variable y (i.e., 3) by entering 3 through the keyboard. Therefore:

sum = *p + *q = 4 + 3 = 7 (which is outputted on the screen).

C++ program to print the product, subtraction and division of two numbers using pointers

```
#include<iostream>
using namespace std;
int main ()
{
int x, y, *p, *q, product, subtract, div;
cout<<"Enter any number:";
cin>> x;
cout<<"Enter any number:";
cin>> y;
p = & x;
q = & y;
product = *p * *q;
subtract = *p - *q;
div= *p / *q;
cout<<"\n product of entered numbers = "<< product;
cout<<"\n subtract of entered numbers = "<< subtract;
cout<<"\n division of entered numbers = "<< div;
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter the number 4
Enter any number:
If you enter the number 2
product of entered numbers = 8
subtract of entered numbers = 2
division of entered numbers = 2
```

will be displayed on the screen.

C++ program to find the greatest of two numbers using pointers

```
#include<iostream>
using namespace std;
int main ()
{
int x, y, *p, *q;
cout<<"Enter any integer:";
cin>> x;
cout<<"Enter any integer:";
cin>> y;
p =    &   x;
q =    &   y;
if (*p>*q)
{
cout<<"x is greater than y";
}
else
{
cout<<"y is greater than x";
}
return 0;
}
```

The output on the screen:

```
Enter any integer:
If you enter the integer 10
Enter any integer:
If you enter the integer 16
y is greater than x
```

will be outputted on the screen.

What is the output of the following programs:

A)

```
#include <iostream>
using namespace std;
int main ()
{
int x;
x=12;
cout<<"per = "<< x;
return 0;
}
```

Answer:

per=12

B)

```
#include <iostream>
using namespace std;
int main ()
{
int x, t, c;
x=12;
t=2;
c = x/t;
cout<<"velocity = "<< c <<"m/s";
return 0;
}
```

Answer:

velocity = 6 m/s

Program 7.4

C++ program to print the sum of two numbers using functions

```
#include<iostream>
using namespace std;
int addition ();
int main ()
{
int answer;
answer = addition ();
cout<<"The sum of two numbers is: "<<answer;
return 0;
}
int addition ()
{
int x, y;
cout<<"Enter any integer:";
cin>>x;
cout<<"Enter any integer:";
cin>>y;
return x+y;
}
```

The output on the screen:

```
Enter any integer:
If you enter the integer 3
Enter any integer:
If you enter the integer 5
sum of two numbers = 8
```

will be displayed on the screen.

int addition (); // the statement implies function declaration

int means integer and int addition () implies: addition () should return integer value.

int addition ()// implies: the function to add the entered values (i.e., 3 and 5) and return the result (i.e., 3 + 5 i.e., 8) to the statement:

```
cout<<"The sum of two numbers is: "<<answer;
```

to make provision to display the output:

```
sum of two numbers = 8
{
int x, y;
cout<<"Enter any integer:";
cin>>x;
cout<<"Enter any integer:";
cin>>y;
return x+y;
}// implies: the body of the function int addition ()
answer = addition (); // implies: the function call i.e., calls the function:
addition ()
```

to add the entered values (i.e., 3 and 5) and return the result (i.e., 3 + 5 i.e., 8)

to the statement:

```
cout<<"The sum of two numbers is: "<<answer;
```

to make provision to display the output:

```
sum of two numbers = 8
```

C++ program to print the product of two numbers using functions

```
#include<iostream>
using namespace std;
int multiplication ();
int main ()
{
int answer;
answer = multiplication ();
cout<<"The product of two numbers is: "<<answer;
return 0;
}
int multiplication ()
{
int x, y;
cout<<"Enter any integer:";
cin>>x;
cout<<"Enter any integer:";
cin>>y;
return x*y;
}
```

The output on the screen:

Enter any integer:

If you enter the integer 3

Enter any integer:

If you enter the integer 5

product of two numbers = 15

will be outputted on the screen.

C++ program to print the greatest of two numbers using functions

```
#include<iostream>
using namespace std;
int largest ();
int main ()
{
int answer;
answer = largest ();
cout<<"The largest of two numbers is: "<<answer;
return 0;
}
int largest ()
{
int x, y;
cout<<"Enter any integer:";
cin>>x;
cout<<"Enter any integer:";
cin>>y;
if (x>y)
return x;
if (y>x)
return y;
}
```

The output on the screen:

Enter any integer:

If you enter the integer 3

Enter any integer:

If you enter the integer 5

largest of two numbers= 5

will be outputted on the screen.

C++ program to print the greatest of three numbers using functions

```
#include<iostream>
using namespace std;
int largest ();
int main ()
{
int answer;
answer = largest ();
cout<<"largest of three numbers= "<< answer;
return 0;
}
int largest ()
{
int x, y, z;
cout<<"Enter any integer:";
cin>>x;
cout<<"Enter any integer:";
cin>>y;
cout<<"Enter any integer:";
cin>>z;
if (x>y & & x>z)
return x;
if (y>x & & y > z)
return y;
if (z>x & & z>y)
return z;
}
```

The output on the screen:

```
Enter any integer:
If you enter the integer 3
Enter any integer:
If you enter the integer 5
Enter any integer:
If you enter the integer 10
largest of three numbers = 10
```

will be outputted on the screen.

C++ program to print the square of the number using functions

```
#include<iostream>
using namespace std;
int square (;
int main ()
{
int answer;
answer = square ();
cout<<"square of the given number = "<< answer;
return 0;
}
int square ()
{
int x;
cout<<"Enter any integer:";
cin>>x;
return x*x;
}
```

The output on the screen:

```
Enter any integer:
If you enter an integer 5
square of the number = 25
```

will be outputted on the screen.

What is the output of the following program:

```
#include<iostream>
using namespace std;
int main ()
{
int x;
x=6;
cout<<"The address of x = "<< & x;
return 0;
}
```

Answer:

```
The address of x = 0x7ffd80d2c06c
```

Program 7.5

Switch (case) allows to make decision from the number of choices i.e., from the number of cases

For example:

```
#include<iostream>
using namespace std;
int main ()
{
char ch;
cout<<"Enter any character:";
cin>>ch;
switch (ch)
{
case 'R':
cout<<"Red";
break;
case 'W':
cout<<"White";
break;
case 'Y':
cout<<"Yellow";
break;
case 'G':
cout<<"Green";
break;
default:
cout<<"Error";
break;
}
return 0;
}
```

The output on the screen:

```
Enter any character:
If you enter a character R
Red
```

will be outputted on the screen.

switch (ch) allow to make decision from the number of choices i.e., from the number of cases

```
case 'R':
case 'W':
case 'Y':
case 'G':
```

Since we have entered the character R (which corresponds to case 'R':)

The statement

```
cout<<"Red";
```

is executed to display the output:

```
Red
```

on the screen.

Suppose you enter a character K

Then the output on the screen is:

```
Error
```

(Entered character K does not correspond to any of the cases:

```
case 'R':
case 'W':
case 'Y':
case 'G':
```

Therefore the statement:

```
cout<<"Error";
```

is executed to display the output:

```
Error
```

on the console screen).

If the statements:

```
case 'R':
cout<<"Red";
```

```
break;
case 'W':
cout<<"White";
break;
case 'Y':
cout<<"Yellow";
break;
case 'G':
cout<<"Green";
break;
default:
cout<<"Error";
break;
```

are replaced by the statements:

```
case 'R':
cout<<"Red";
case 'W':
cout<<"White";
case 'Y':
cout<<"Yellow";
break;
case 'G':
cout<<"Green";
break;
default:
cout<<"Error";
break;
```

Then the output on the screen is:

```
Red
White
Yellow
```

i.e., the output will be printed till yellow even though you have entered the character R.

Program 7.6

C++ program to print the output:

```
Element [0] = 16
Element [1] = 18
Element [2] = 20
Element [3] = 25
Element [4] = 36
```

using arrays:

```
#include<iostream>
using namespace std;
main ()
{
int i;
int num [5] = {16, 18, 20, 25, 36};
for (i=0; i<5; i++)
cout<<"Element ["<< i <<" ] = "<< num [i] << endl;
return 0;
}
```

The output on the screen:

```
Element [0] = 16
Element [1] = 18
Element [2] = 20
Element [3] = 25
Element [4] = 36
```

The statement:

int num [5] = {16, 18, 20, 25, 36};

imply that we are creating an integer array (and the name of array is num) consisting of 5 values (i.e., 16, 18, 20, 25, 36) of the same data type int.

The number of values between the braces { } cannot be larger than the number of values that we declare for the array between square brackets [ ].

There are 5 integers i.e., 16, 18, 20, 25, 36 within the braces { }, so 5 is written within the square brackets [ ].

If there were 6 integers i.e., 16, 18, 20, 25, 36, 42 within the braces { }, then 6 must be written within the square brackets [ ].

Note: With the declaration int num [5], computer creates 5 memory cells with name num [0], num [1], num [2], num [3], num [4].

And since:

int num [5] = {16, 18, 20, 25, 36};

the values 16, 18, 20, 25, 36 are stored in num [0], num [1], num [2], num [3], num [4] respectively.

How the execution takes its way through the for loop statement

i=0

Is i<5 true?

Yes, print this

Element [0] = 16

using the statement:

cout<<"Element ["<< i <<"] = "<< num [i] << endl;

format string %d in the square brackets indicates that the value to be displayed at that point in the string i.e., with the square brackets [ ] needs to be taken from a variable (which is i i.e., i=0) and the format string %d after the statement (\n Element [%d] =) indicates that the value to be displayed at that point in the string i.e., after the statement (\n Element [%d] =) needs to be taken from a variable (which is stored in num [i] i.e., num [0] i.e., 16).

Now,

i=1

Is i<5 true?

Yes, print this

Element [1] = 18

using the statement:

cout<<"Element ["<< i <<"] = "<< num [i] << endl;

format string %d in the square brackets indicates that the value to be displayed at that point in the string i.e., with the square brackets [ ] needs to be taken from a variable (which is i i.e., i=1) and the format string %d after the statement (\n Element [%d] =) indicates that the value to be displayed at that point in the string i.e., after the statement (\n Element

[%d] =) needs to be taken from a variable (which is stored in num [i] i.e., num [1] i.e., 18).

Now,

i=2

Is i<5 true?

Yes, print this

Element [2] = 20

using the statement:

cout<<"Element ["<< i <<"] = "<< num [i] << endl;

format string %d in the square brackets indicates that the value to be displayed at that point in the string i.e., with the square brackets [ ] needs to be taken from a variable (which is i i.e., i=2) and the format string %d after the statement (\n Element [%d] =) indicates that the value to be displayed at that point in the string i.e., after the statement (\n Element [%d] =) needs to be taken from a variable (which is stored in num [i] i.e., num [2] i.e., 20).

Now,

i=3

Is i<5 true?

Yes, print this

Element [3] = 25

using the statement:

cout<<"Element ["<< i <<"] = "<< num [i] << endl;

format string %d in the square brackets indicates that the value to be displayed at that point in the string i.e., with the square brackets [ ] needs to be taken from a variable (which is i i.e., i=3) and the format string %d after the statement (\n Element [%d] =) indicates that the value to be displayed at that point in the string i.e., after the statement (\n Element [%d] =) needs to be taken from a variable (which is stored in num [i] i.e., num [3] i.e., 25).

Now,

i=4

Is i<5 true?

Yes, print this

Element [4] = 36

using the statement:

cout<<"Element ["<< i <<"] = "<< num [i] << endl;

Stop because the condition i<5 is achieved.

format string %d in the square brackets indicates that the value to be displayed at that point in the string i.e., with the square brackets [ ] needs to be taken from a variable (which is i i.e., i=4) and the format string %d after the statement (\n Element [%d] =) indicates that the value to be displayed at that point in the string i.e., after the statement (\n Element [%d] =) needs to be taken from a variable (which is stored in num [i] i.e., num [4] i.e., 36).

Suppose the statement:

cout<<"Element ["<< i <<"] = "<< num [i] << endl; is replaced by the statement:

cout<<"Element ["<< i <<"] = "<< num [0] << endl;

Then the output on the screen:

Element [0] = 16
Element [1] = 16
Element [2] = 16
Element [3] = 16
Element [4] = 16

Suppose the statement:

cout<<"Element ["<< i <<"] = "<< num [i] << endl; is replaced by the statement:

cout<<"Element ["<< i <<"] = "<< num [1] << endl;

The output on the screen:

Element [0] = 18
Element [1] = 18
Element [2] = 18
Element [3] = 18
Element [4] = 18

Suppose the statement:

cout<<"Element ["<< i <<"] = "<< num [i] << endl; is replaced by the statement:

cout<<"Element ["<< i <<"] = "<< num [2] << endl;

The output on the screen:

Element [0] = 20
Element [1] = 20
Element [2] = 20
Element [3] = 20
Element [4] = 20

Suppose the statement:

cout<<"Element ["<< i <<"] = "<< num [i] << endl; is replaced by the statement:

cout<<"Element ["<< i <<"] = "<< num [3] << endl;

The output on the screen:

Element [0] = 25
Element [1] = 25
Element [2] = 25
Element [3] = 25
Element [4] = 25

Suppose the statement:

cout<<"Element ["<< i <<"] = "<< num [i] << endl; is replaced by the statement:

cout<<"Element ["<< i <<"] = "<< num [4] << endl;

The output on the screen:

Element [0] = 36
Element [1] = 36
Element [2] = 36
Element [3] = 36
Element [4] = 36

If the condition:

i<5

is replaced by the condition:

i<=5

Then the output on the screen is:

```
Element [0] = 16
Element [1] = 18
Element [2] = 20
Element [3] = 25
Element [4] = 36
Element [5] = 3656
```

3656 is the number stored in the memory i.e., any number stored in the memory will be displayed.

If the statement:

int num [5] = {16, 18, 20, 25, 36}; is replaced by the statement:

int num [i] = {16, 18, 20, 25, 36};

Then the compilation will be displayed on the screen because there are 5 elements within the braces {} not i elements.

Note:

C++ program to print the sum of the elements in array.

```
#include<iostream>
using namespace std;
int main ()
{
int i, sum = 0;
int num [5] = {16, 18, 20, 25, 36};
for (i=0; i<5; i++)
sum = sum + num [i];
cout<<"Sum of the Elements in the array = "<< sum;
return 0;
}
```

The output on the screen:

```
Sum of the Elements in the array = 115
```

i.e., 16 + 18 + 20 + 25 + 36 = 115

How the Execution takes its way through the for loop statement

i=0 (sum = 0)

Is i<5 true?

Yes, do this

sum = sum + num [i] = sum + num [0] = 0 +16 =16

Now,

i=1 (sum = 16)

Is i<5 true?

Yes, do this

sum = sum + num [i] = sum + num [1] = 16 +18 =34

Now,

i=2 (sum = 34)

Is i<5 true?

Yes, do this

sum = sum + num [i] = sum + num [2] = 34 +20 =54

Now,

i=3 (sum = 54)

Is i<5 true?

Yes, do this

sum = sum + num [i] = sum + num [3] = 54 +25 =79

Now,

i=5 (sum = 79)

Is i<5 true?

Yes, do this

sum = sum + num [i] = sum + num [5] = 79 + 36 =115

stop because the condition i<5 is achieved

The statement:

cout<<"Sum of the Elements in the array = "<< sum; is executed to display the output:

Sum of the Elements in the array = 115

on the screen.

If the statement:

int i, sum = 0;

is replaced by int i, sum = 1;

Then The output on the screen:

Sum of the Elements in the array = 116

C++ program to print the average of the elements in array

```
#include<iostream>
using namespace std;
int main ()
{
int i, avg, sum = 0;
int num [5] = {16, 18, 20, 25, 36};
for (i=0; i<5; i++)
sum = sum + num [i];
avg = sum/5;
cout<<"Sum of the Elements in the array = "<< sum;
cout<<"average of the elements in the array= "<< avg;
return 0;
}
```

The output on the screen:

```
Sum of the Elements in the array = 115
average of the elements in the array = 23
```

Write a program to print:

```
Einstein [0] = E
Einstein [1] = I
Einstein [2] = N
Einstein [3] = S
Einstein [4] = T
Einstein [5] = E
Einstein [6] = I
Einstein [7] = N
```

using arrays

Answer:

```
#include<iostream>
using namespace std;
int main ()
{
```

```
int i;
char name [8] = {'E', 'I', 'N', 'S', 'T', 'E', 'I', 'N'};
for (i=0; i<8; i++)
cout<<"Element ["<< i <<" ] = "<< name [i] << endl;
return 0;
}
```

What will be the output of the following programs?

i)

```
#include <iostream>
#include <math.h>
using namespace std;
int main ()
{
cout<<""<< cbrt (27);
return 0;
}
```

Answer:

```
3
```

ii)

```
#include <iostream>
using namespace std;
int main ()
{
char i;
char body [4] = {'b', 'o', 'd', 'y'};
for (i=0; i<4; i++)
cout<<"\n body ["<<body [i] <<" ] = "<< body [i] << endl;
return 0;
}
```

Answer:

```
body [b] = b
body [o] = o
body [d] = d
body [y] = y
```

iii)

```
#include <iostream>
#include <malloc.h>
using namespace std;
int main ()
{
int x=2;
cout<<""<< malloc (200*sizeof (x));
return 0;
}
```

Answer:

```
8183824
```

Program 7.7

C++ program to print the output:

```
Name of the book = B
Price of the book = 135.00
Number of pages = 300
Edition = 8
```

using structures

```
#include<iostream>
using namespace std;
int main ()
{
struct book {
char name;
float price;
int pages;
int edition;
};
struct book b1= {'B', 135.00, 300, 8};
cout<<"Name of the book = "<< b1.name<< endl;
cout<<"Price of the book = "<< b1.price<<endl;
cout<<"Number of pages = "<< b1.pages<<endl;
cout<<"Edition of the book = "<< b1.edition<< endl;
return 0;
}
```

The output on the screen:

```
Name of the book = B
Price of the book = 135.00
Number of pages = 300
Edition of the book = 8
```

The statement:

```
struct book {
char name;
float price;
int pages;
int edition;
};
```

imply the structure definition i.e., we are defining a structure (and the data type name of the structure is book) and it consists of elements:

name (which is of data type char), price (which is of data type float), pages (which is of data type int), edition (which is of data type int) which are placed within the body of the structure.

The statement:

```
struct book b1;
```

imply the structure variable declaration (where b1 denote the structure variable)

Why structure variable b1 is declared or defined?

In order to assign the values to the elements within the body of the structure, each element must be linked with structure variable with dot operator or period operator or member accessibility operator.

For example: name is the element which must be linked with structure variable b1 with dot operator to assign a value B to the element "name".

The statement:

```
cout<<"Name of the book = "<< b1.name<< endl;
```

is executed to print the output:

```
Name of the book = B
```

on the screen.

The statement:

```
cout<<"Price of the book = "<< b1.price<<endl;
```

is executed to print the output:

```
Price of the book = 135.00
```

on the screen.

The statement:

```
cout<<"Number of pages = "<< b1.pages<<endl;
```

is executed to print the output:

```
Number of pages = 300
```

on the screen.

The statement:

```
cout<<"Edition of the book = "<< b1.edition<< endl;
```

is executed to print the output:

```
Edition of the book = 8
```

on the screen.

What will be output of the following programs?

A)

```
#include<iostream>
using namespace std;
struct book {
char name;
float price;
int pages;
int edition;
};
int main ()
{
struct book b1;
b1.name = 'C';
b1.price = 135.00;
b1.pages = 300;
b1.edition = 8;
cout<<"Name of the book = bulgarian "<< b1.name << endl;
cout<<"\n Price of the book = "<< b1.price;
cout<<"\n Number of pages = "<< b1.pages<<endl;
cout<<"\n Edition of the book = "<< b1.edition;
}
```

Answer:

```
Name of the book = B
Price of the book = 135.000000
Number of pages = 300
Edition of the book = 8
```

B)

```
#include <iostream>
using namespace std;
int main ()
{
for (;; ) {
cout<<"This loop will run forever.\n";
}
return 0;
}
```

Answer:

```
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever.......... continues
```

Program 7.8

Continue and break statements:

i)

```
#include <iostream>
using namespace std;
int main ()
{
int i;
for (i=1; i<=5; i++)
{
if (i==3)
{
continue;
}
cout<<"\n "<< i;
}
return 0;
}
```

Output on the screen:

```
1
2
4
5
```

ii)

```
#include <iostream>
using namespace std;
int main ()
{
int i;
for (i=1; i<=5; i++)
{
if (i==3)
{
break;
}
cout<<"\n "<< i;
}
return 0;
}
```

Output on the screen:

```
1
2
```

Program 7.9

C++ program to convert the upper case letter to lower case letter

```
#include<iostream>
using namespace std;
```

```
int main ()
{
char ch = 'A';
char b = tolower (ch);
cout<<" upper case letter "<< ch <<" is converted to lower case letter "<< b;
return 0;
}
```

Output on the screen:

```
upper case letter A is converted to lower case letter a
```

If you want to enter the character through the keyboard, then the above program should take the form:

```
#include<iostream>
using namespace std;
int main ()
{
char ch;
cout<<"Enter any character:";
cin>>ch;
char b = tolower (ch);
cout<<" upper case letter "<< ch <<" is converted to lower case letter "<< b;
return 0;
}
```

Output on the screen:

```
Enter any character:
```

If you enter the character C

```
upper case letter C is converted to lower case letter c
```

will be outputted on the screen.

Program 8.0

C++ program to convert the lower case letter to upper case letter

```
#include<iostream>
using namespace std;
int main ()
{
char ch = 'a';
char b = toupper (ch);
cout<<" lower case letter "<<ch<<" is converted to upper case letter "<<b;
return 0;
}
```

Output on the screen:

```
lower case letter a is converted to upper case letter A
```

If you want to enter the character through the keyboard, then the above program should take the form:

```
#include<iostream>
using namespace std;
int main ()
{
char ch;
cout<<"Enter any character:";
cin>>ch;
char b = toupper (ch);
cout<<" lower case letter "<<ch<<" is converted to upper case letter "<<b;
return 0;
}
```

Output on the screen:

```
Enter any character:
```

If you enter the character h

```
lower case letter h is converted to upper case letter H
```

will be outputted on the screen.

Program 8.1

C++ program to test whether the entered character is upper case letter or not

```
#include<iostream>
using namespace std;
int main ()
{
char ch = 'a';
if (isupper (ch))
cout<<"you have entered the upper case letter";
else
cout<<"you have entered the lower case letter";
return 0;
}
```

Output on the screen:

```
you have entered the lower case letter
```

If the statement:

char ch = 'a'; is replaced by the statement:

char ch = 'A';

Then the output on the screen is:

```
you have entered the upper case letter
```

Program 8.2

C++ program to test whether the entered character is lower case letter or not

```
#include<iostream>
using namespace std;
int main ()
{
char ch = 'a';
if (islower (ch))
cout<<"you have entered the lower case letter";
else
cout<<"you have entered the upper case letter";
return 0;
}
```

Output on the screen:

```
you have entered the lower case letter
```

Program 8.3

C++ program to print the value of tan inverse x (i.e., the value of tan-1 x)

```
#include<iostream>
#include<math.h>
```

```
using namespace std;
int main ()
{
int x = 20;
cout<<"the value of tan inverse x = "<< atan (x);
return 0;
}
```

Output on the screen:

```
the value of tan inverse x = 1.520838
```

Program 8.4

C++ program to print the value of tan inverse x/y (i.e., the value of tan-1x/y)

```
#include<iostream>
#include<math.h>
using namespace std;
int main ()
{
int x,y;
x = 20;
y =20;
cout<<"the value of tan inverse x/y = "<< atan2(x,y);
return 0;
}
```

Output on the screen:

```
the value of tan inverse x/y = 0.785398
```

Program 8.5

C++ program to print the value of fmod (x, y)

```
#include<iostream>
#include<math.h>
using namespace std;
int main ()
{
float x = 20.500000;
float y =20.799999;
cout<<" the remainder of "<<x <<" divided by "<<y <<" is: "<< fmod (x,y);
return 0;
}
```

Output on the screen:

```
the remainder of 20.500000 divided by 20.799999 is 20.500000
```

Program 8.6

C++ program to print the value of ~x

```
#include<iostream>
using namespace std;
int main ()
{
int x, y;
x = 205;
y=~x;
cout<<"the value of y is: "<< y;
return 0;
}
```

Output on the screen:

```
the value of y is:-206
```

If the statement:

y=~x; is replaced by the statement:

y= -(~x);

Then the output on the screen is:

```
the value of y is:206
```

What will be the output of the following programs:

i)

```
#include<iostream>
using namespace std;
int main ()
{
int i = 54;
int y = i<<1;
cout<<"The value of y = "<< y;
return 0;
}
```

Answer:

The value of y = 108

If the statement:

i<<1 is replaced by the statement: i<<2

Then the output on the screen is:

The value of y = 216

Note:

i<<1 implies 54 * 2 = 108
i<<2 implies 54 * 4 = 216
i<<3 implies 54 * 6 = 324
i<<4 implies 54 * 8 = 432

ii)

```
#include<iostream>
using namespace std;
int main ()
{
int i = 54;
int y = i>>1;
cout<<"The value of y = "<< y;
return 0;
}
```

Answer:

The value of y = 27

If the statement:

i>>1 is replaced by the statement: i>>2

Then the output on the screen is:

The value of y = 13

Note:

i>>1 implies 54 / 2 = 27
i>>2 implies 54 / 4 = 13
i>>3 implies 54 / 6 = 9
i>>4 implies 54 / 8 = 6

<< implies: left shift operator
>> implies: right shift operator

Program 8.7

C++ program to print the length of the entered character (i.e., to print the length of the string)

```
#include<iostream>
#include<string.h>
using namespace std;
int main ()
{
char ch [4];
cout<<"Enter any word: ";
cin>>ch;
cout<<"The length of the string = "<< strlen (ch);
return 0;
}
```

Output on the screen:

```
Enter any word:
```

If you enter the word dog

```
The length of the string = 3
```

will be displayed on the console screen because there are three letters in the word dog.

Suppose if you enter the word tech

```
The length of the string = 4
```

will be displayed on the console screen because there are four letters in the word tech.

Program 8.8

C++ program to print the factorial of the entered number

```
#include<iostream>
using namespace std;
int main ()
{
int i, n, fact=1;
cout<<"Enter any number:";
cin>>n;
for (i=1; i<=n; i++)
fact = fact *i;
cout<<"\n Entered number is: "<< n;
cout<<"\n The factorial of the entered number"<<n<<"is:"<< fact;
return 0;
}
```

Output on the screen:

```
Enter any number:
```

If you enter the number 2

```
Entered number is: 2
The factorial of the entered number 2 is: 2
```

will be displayed on the screen.

Suppose if you enter the number 4

```
Entered number is: 4
The factorial of the entered number 4 is: 24
```

will be displayed on the screen.

Java Programming

Java is a high level programming language conceived by James Gosling

, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan at Sun Microsystems, Inc. in 1991 to create programs to control consumer electronics (which is now a subsidiary of Oracle Corporation) and released in 1995, used in internet programming, mobile devices, games, e-business solutions etc., because of its reliability, high performance, simplicity and easy to use and quick to learn and rigid versus extensibility.

Process of Java program execution: A Java program:

```
public class HelloWorld {
public static void main (String [] args) {
System.out.println ("Hello, World!");
}
}
```

is written using Text Editor, such as [Notepad++, Notepad] and saved with [.java] Extension.

File Saved with [.java] extension is called Source Program or Source Code.

```
// ---HelloWorld.java-------------------------------------------
---------------------------------------------
public class HelloWorld {
public static void main (String [] args) {
System.out.println ("Hello, World!");
}
}
-------(because the class name is HelloWorld the source file should be named as HelloWorld.java)---------------//
```

and sent to the java compiler (i.e., javac compiler) where the source program is compiled (i.e., the program is entirely read and translated into Java byte codes (but not into machine language)).

If the javac compiler finds any error during compilation, it provides information about the error to the programmer. The programmer has to review code and check for the solution.

And if there are no errors the translated program (i.e., java byte codes -- a highly optimized set of instructions) is stored in computers main memory as HelloWorld.class and since the java byte codes cannot be trusted to be correct. Therefore before execution they are verified and converted to machine level language i.e., machine code sequence of 0s and 1s by Java run-time system, which is called the Java Virtual Machine (JVM) and is executed by a Java interpreter and

```
Hello, World!
```

is displayed on the console screen.

Note:

JVM (Java Virtual Machine) resides under

RAM (Random Access Memory the stuff that boost up your computer to run faster and allows your computer to perform many tasks at the same time)

and JVM comprises:

Class Loader: it loads.class file that contains Java byte codes.

Byte Code Verifier: it verifies byte codes.

Execution Engine: it translates java byte codes to machine codes and executes them.

In the statement:

public class HelloWorld

The word "HelloWorld" implies: name of the class is HelloWorld and this class is public.

public means that the class HelloWorld can be accessed by any other class in any package.

In the program:

```
//-----------------------------------------------------------------------------------------
public class HelloWorld {
public static void main (String [] args) {
System.out.println ("Hello, World!");
}
}
-----------------------------------------------------------------------------------------//
```

public class HelloWorld {

} // imply the body of the class HelloWorld (Here: the curly brace '{' imply the beginning of the class and the curly brace '}' imply the end of the class) within which the

main method

public static void main (String [] args) {

} is written.

public static void main (String [] args) imply: main method (a collection of statements or methods like System.out.println ( ) that are grouped together to perform an operation)

and this main method is public

and

{

} imply the body of the main method

(Here: the curly brace '{' imply the beginning of the main method and the curly brace '}' imply the end of the main method) within which the statement:

System.out.println ("Hello, World!");

is written and executed.

Note: main method in java functions like main function main () in C and C++.

If the statement:

public class HelloWorld is replaced by the statement:

public class sample i.e.,

```
public class sample {
public static void main (String [] args) {
System.out.println ("Hello, World!");
}
}
```

Then the error will be displayed on the screen because the program written in notepad is saved as HelloWorld.java not as sample.java.

If we want to write the statement:

public class sample instead of the statement:

public class HelloWorld, then we have to save the program written in notepad as sample.java but not as HelloWorld.java.

Like C & C++, Java is also a case sensitive language i.e., capital letters (or upper case letters) must be avoided to prevent the display of error on the screen

For example: If the statement:

PUBLIC static void main (String [] args) is written instead of the statement:

public static void main (String [] args), compilation Error will be displayed on the screen.

If we forget to end each program statement within the body of main method with a semicolon (";"), Error will be displayed on the screen

public static void main (String [] args) The program begins its execution with the method:

public static void main (String [] args)- the main method -- the entry point of the program execution

i.e., the point from where the execution of Java program begins.

Semicolon: program is a set of instructions and each instruction (or each statement) is ended by a semicolon. Like in an English paragraph each sentence is ended by a full stop which tells that one sentence ends and another begins, semicolon implies that one instruction (or statement) ends and another begins.

In the statement:

System.out.println ();

System name of a standard class that contains variables and methods for supporting simple keyboard and character output to the display.

out represents the standard output stream println () output method of the Java language which makes provision to print the output in the next line:

Hello,world!

on the screen.

The text

Hello,world!

should be enclosed by the double quotation marks ("") and should be written within the println method and this

println method should be ended with the semicolon i.e.,

System.out.println ("Hello,world!");

otherwise the compilation error will be displayed on the console screen.

//--

```
-----------------------------------------------------------------------------
---------------
public class HelloWorld {
public static void main (String [] args) {
System.out.println ("Hello, World!");
System.out.println ("Hello, World!");
}
}
```

Output on the screen:

```
Hello, World!
Hello, World!
```

```
public class HelloWorld {
public static void main (String [] args) {
System.out.print ("Hello, World!");
System.out.print ("Hello, World!");
}
}
```

Output on the screen:

```
Hello, World!Hello, World!
----------------------------------------------------------------------
-----------------------------------------------------------------------------
-------------//
```

In the statement:

public static void main (String [] args)

public implies: this method can accessed from anywhere outside the class HelloWorld

If the word "public" in the statement:

public static void main (String [] args)

is replaced by the word

private

or

protected

Then compilation error will be flagged on the screen because if the method is declared private or protected then this method does not make itself available to JVM for execution.

main implies the name of the method

static means the main method is the part of the class HelloWorld

Why static?

Because the program execution begins from the main method and if the main method is not declared static then the execution of the program does not take place.

void implies the main method does not return any value i.e., main method return nothing when it completes execution.

String args [] While running the program if we want to pass something to the main method, then this parameter is used as the way of taking input from the user--so we can pass some strings while running the program if we want.

Moreover, JVM cannot recognize the method:

public static void main (String [] args)

as method if the parameter String [] args is not included.

If the word args in the statement:

public static void main (String [] args) is replaced by another word say jamesgosling or java i.e.,

```
//-----------------------------------------------------------------------
--------------------------------------------------------------------------
public class HelloWorld
{
public static void main (String [] jamesgosling)
{
System.out.println ("Hello, World!");
}
}
public class HelloWorld {
public static void main (String [] java) {
System.out.println ("Hello, World!");
}
}
----------------------------------------------------------------------
------------------------------------------------------------------------//
```

No compilation error will be displayed on the screen i.e., Hello, World! will be outputted on the screen without display of any error on the screen.

If the statement:

public static void main (String [] args)

is replaced by the statement public static void main (String []) -- Then the error is displayed on the screen.

Note: Most Java programmers prefer args and argv i.e., the statements:

public static void main (String [] args)

and public static void main (String [] argv) are preferred.

If the space is left between the words Hello and World i.e., if the statement:

public class Hello World is written instead of the statement:

public class HelloWorld. Then the compilation error will be displayed on the screen.

Program 1.1

Java program to print the word "hello Bill Gates" on screen

```
public class HelloWorld {
public static void main (String [] args) {
System.out.println ("hello Bill Gates");
}
}
```

The output on the screen:

```
hello Bill Gates
```

Program 1.2

Java program to print the word "****hello silicon city****" on screen

```
public class HelloWorld {
public static void main (String [] args) {
System.out.println (" ****hello silicon city****
");
}
}
```

The output on the screen:

```
****hello silicon city****
```

Program 1.3

Java program to print

```
*
*****
*****
*****
*****
```

on screen

```
public class HelloWorld {
public static void main (String [] args) {
System.out.println ("\n     *      ");
System.out.println ("\n ***** ");
System.out.println ("\n ***** ");
System.out.println ("\n ***** ");
System.out.println ("\n ***** ");
}
}
```

The output on the screen:

```
*
*****
*****
*****
*****
```

If new line \n is not included in the above program then the output on the screen is:

```
*********************
```

Write a program to print the following outputs:

(a)

```
*
****
**java**
****
*
```

(b)

```
****************
* *
* Hello World! *
* *
****************
```

(c)

```
Braces come in pairs!
Comments come in pairs!
All statements end with a semicolon!
Spaces are optional!
Must have a main method!
java is done mostly in lowercase. Like C    &
C++ its also a case-sensitive language
```

Answers:

a)

```
public class HelloWorld {
public static void main (String [] args) {
System.out.println ("\n     *      ");
System.out.println ("\n     ****");
System.out.println ("\n     **java** ");
System.out.println ("\n     ****        ");
System.out.println ("\n     *      ");
}
}
```

b)

```
public class HelloWorld {
public static void main (String [] args) {
System.out.println ("\n     ****************
");
System.out.println ("\n     * *    ");
System.out.println ("\n     *   Hello   World!   *
");
System.out.println ("\n     * *    ");
System.out.println ("\n     ****************
");
}
}
```

c)

```
public class HelloWorld {
public static void main (String [] args) {
System.out.println ("\n Braces come in pairs!");
System.out.println ("\n Comments come in
pairs!");
System.out.println ("\n All statements end with
a semicolon!");
System.out.println ("\n Spaces are optional!");
System.out.println ("\n Must have a main
method!");
System.out.println ("\n java is done mostly in
lowercase. Like C    &    C++    it's    also    a
case-sensitive language");
}
}
```

Program 1.4

Java program to find the area of the circle

```
public class HelloWorld {
public static void main (String [] args) {
int r, area;
r = 2;
area = 4 * 3.14 * r * r;
System.out.println ("The area of the circle = " +
area);
}
}
```

The output on the screen:

```
The area of the circle = 50
```

int means the integer data type.

Note: An integer is a whole number - no fractions, decimal parts, or funny stuff.

The statement:

int r, area; imply that we are creating the integer variables r, area.

Equal sign implies: storage operator.

The statements:

```
r = 2;
area = 4 * 3.14 * r * r;
```

imply that we are storing the values to the created variables (i.e., we are storing the value 2 for r and 4 * 3.14 * r * r for area).

Comma in the statement:

int r, area; imply:: variable separator.

If multiplication sign & times; is used instead of multiplication operator * i.e.,

```
area = 4   3.14 r     r;
```

then the compilation error will be displayed on the screen.

In C language, the statement:

```
printf ("The area of the circle = %d ", area);
```

make the provision to print the output on the screen.

In C++ language, the statement

```
cout<<"The area of the circle = "<< area;
```

make the provision to print the output on the screen.

whereas in the Java language, the statement:

```
System.out.println ("The area of the circle = " + area);
```

make the provision to print the output on the screen.

In the statement:

```
System.out.println ("The area of the circle = " + area);
```

There are two strings:

The area of the circle =

area

plus operator (+) functions as the concatenation operator (concatenation means connecting two statements to produce a single statement) which (here) concatenates

the string:

"The area of the circle = " and the string:

"area (which is 4 * 3.14 * r * r (=50 since r = 2))" -- producing a String statement:

```
The area of the circle = 50
```

which will be displayed on the screen as the result.

The area of the circle is = 50. 24 (for r = 2) but

The area of the circle = 50 is displayed on the screen because the data type int is used instead of data type float.

If the data type float is used instead of int i.e., then the output on the screen is:

```
The area of the circle = 50.24
```

If you write:

4 * 3.14 * r^2; instead of 4 * 3.14 * r * r;

Then error is displayed on the screen because like in other high level languages (such as C and C++) there is no operator for performing r ^ 2 operation so the statement:

4 * 3.14 * r^2; is invalid.

Even though if we write ARGS instead of args i.e., even though if we express args in capital letter, No error will be displayed on the screen.

public static void main (String [] ARGS) no error will be displayed on the console screen.

Program 1.5

Java program to find the circumference of the circle

```
public class HelloWorld {
public static void main (String [] args) {
float r, circumference;
r = 2;
circumference = 3.14 * r * r;
System.out.println ("The circumference of the circle = " + circumference);
}
}
```

The output on the screen is:

```
The circumference of the circle = 12.56
```

What will be the output of the following programs:

a)

```
public class HelloWorld {
public static void main (String [] args) {
double l, b, area;
l=2;
b=2.5;
area = l*b;
System.out.println ("The area of the rectangle = " + area);
}
}
```

Answer:

The area of the rectangle = 5.0

b)

```
public class HelloWorld {
public static void main (String [] args) {
int a, b, c;
a= 3;
b=3;
c=3;
if ((a + b< c) || (b + c < a) || (a==b   &   & b==c))
System.out.println (" the triangle is equilateral");
else
System.out.println (" the triangle is not possible");
}
}
```

Answer:

the triangle is equilateral

Program 1.6

Java program to convert the temperature in Celsius to Fahrenheit

```
public class HelloWorld{
public static void main (String [] args){
float C, F;
C=38.5;
F = 9*C/5 +32;
System.out.println ("temperature in Fahrenheit= " +F);
}
}
```

The output on the screen:

```
temperature in Fahrenheit= 101.3
```

Program 1.7

Java program to find the sum of two numbers

```
public class HelloWorld
{
public static void main (String [] args)
{
int a, b, sum;
a=1;
b=2;
sum = a + b;
System.out.println ("the sum of a and b = " + sum);
}
}
```

The output on the screen:

```
the sum of a and b = 3
```

If you want to supply the values for a and b through the key board, then we have to rewrite the program as follows:

```
import java.util.Scanner;
public class HelloWorld
{
public static void main (String [] args) {
int a, b, sum;
Scanner scan = new Scanner (System.in);
System.out.print ("Enter any two Numbers: ");
a = scan.nextInt ();
b = scan.nextInt ();
sum = a + b;
System.out.println ("the sum of a and b = " + sum);
}
}
```

The output on the screen:

```
Enter any two Numbers:
```

If you enter two numbers 2 and 3

```
the sum of a and b = 5
```

will be outputted on the screen

//---

Scanner is a class found in java.util package. So to use Scanner class, we first need to include: java.util package in our program.

```
import java.util.Scanner; // This will import just the Scanner class
import java.util.*; // This will import the entire java.util package
```

---//

The statement:

```
Scanner scan = new Scanner (System.in);
```

implies: declaring an object of the Scanner class "scan" to read the values entered for a and b through the key board.

And the statements:

```
a = scan.nextInt ();
b = scan.nextInt ();
```

imply: scan is an object of Scanner class and nextInt () is a method of the object "scan" that allows the object "scan" to read only integer values from the keyboard.

//---

Same as the method:

nextInt () that allows the object "scan" to read only integer values from the keyboard, methods that allows the object "scan" to read other data types from the keyboard are listed below:

Methods	datatype
nextInt ()	Integer
nextFloat ()	Float
nextDouble ()	Double
nextLong ()	Long
nextShort ()	Short
next ()	Single word
nextLine ()	Line of Strings
nextBoolean ()	Boolean

---//

Program 1.8

Java program to find the square root of a number

i)

```
public class HelloWorld
{
public static void main (String [] args) {
float x;
x = 233;
System.out.println (" square root of a number = " + Math.sqrt (x));
}
}
```

The output on the screen:

```
square root of a number = 15.264
```

If you want to supply the value for x through the key board, then the above program should take the form:

```
import java.util.Scanner;
public class HelloWorld {
public static void main (String [] args) {
int x;
Scanner scan = new Scanner (System.in);
System.out.print ("Enter any Number: ");
x = scan.nextFloat ();
System.out.println (" square root of a number = " + Math.sqrt (x));
}
}
```

The output on the screen:

Enter any Number:

If you enter the number 233

square root of a number = 15.264337522

will be outputted on the screen.

ii)

```
public class HelloWorld
{
public static void main (String [] args) {
double x;
x = 233;
System.out.println (" square root of a number = " + Math.sqrt (x));
}
}
```

The output on the screen:

```
square root of a number = 15.264337522473747
```

If you want to supply the value for x through the key board, then the above program should take the form:

```
import java.util.Scanner;
public class HelloWorld {
public static void main (String [] args) {
double x;
Scanner scan = new Scanner (System.in);
System.out.print ("Enter any Number: ");
x = scan.nextDouble ();
System.out.println (" square root of a number = " + Math.sqrt (x));
}
}
```

The output on the screen:

Enter any Number:

If you enter the number 233

square root of a number = 15.264337522473747

will be outputted on the screen.

Program 1.9

What will be the output of the following program:

```
public class HelloWorld{
public static void main (String [] args) {
char c;
c = 'A';
System.out.println ("ch= " + c);
}
}
```

The output on the screen:

```
ch=A
```

If you want to supply the value for c through the key board, then the above program should take the form:

```
public class HelloWorld {
public static void main (String [] args) throws Exception {
char c;
System.out.print ("Enter a character:");
c = (char)System.in.read ();
System.out.println ("ch= " + c);
}
}
```

The output on the screen:

Enter a character:

If you enter the character K

ch= K

will be outputted on the screen.

Note: Exception is a problem that arises during the execution of a program.

When an exception occurs, program abnormally terminates and disrupts -

throws Exception should be written after the statement public static void main (String [] args) so that the

exceptions are thrown to the operating system to handle and the program will be successfully executed and the output will be displayed on the screen.

Program 2.0

```
import java.util.Scanner;
public class HelloWorld {
public static void main (String [] args) {
String m;
Scanner in = new Scanner (System.in);
System.out.print ("Enter the word: ");
m = in.nextLine ();
System.out.println (" the word you entered = " + m);
}
}
```

The output on the screen:

Enter the word:

If you enter the word dog

the word you entered = dog

will be outputted on the screen.

//---

Note:

If the statement:

m = scan.nextLine ();

is written instead of

m = in.nextLine ();

Then we have to replace the statement:

Scanner in = new Scanner (System.in);

by the statement:

Scanner scan = new Scanner (System.in);

Otherwise compilation error will be displayed on the screen.

---//

What is the mistake in the following program:

```
public class HelloWorld
{
static public void main (String args []) {
float x;
x = 233;
System.out.println (" cube root of a number = "
+ Math.cbrt (x));
}
}
```

Answer:

There is no mistake in the above program.

The statement:

public static void main (String [] args) can also be written as:

static public void main (String args [])

The output on the screen is:

```
cube root of a number = 6.1534494936636825
```

Program 2.1

Java program to find the product of two numbers.

```
public class HelloWorld{
public static void main (String [] args) {
int a, b, product;
a=1;
b=2;
product = a * b;
System.out.println ("the product of a and b = " +
product);
}
}
```

The output on the screen:

```
the sum of a and b = 2
```

If you want to supply the values for a and b through the key board, then we have to rewrite the above program as follows:

```
import java.util.Scanner;
public class HelloWorld {
public static void main (String [] args) {
int a, b, product;
Scanner scan = new Scanner (System.in);
System.out.print ("Enter any two Numbers: ");
a = scan.nextInt ();
b = scan.nextInt ();
product = a * b;
System.out.println ("the product of a and b = " +
product);
}
}
```

The output on the screen:

```
Enter any two Numbers:
```

If you enter two numbers 6 and 3

```
the product of a and b = 18
```

will be outputted on the screen

Note:

If you want to assign the floating point values for a & b, then the above program should take the form:

```
import java.util.Scanner;
public class HelloWorld {
public static void main (String [] args) {
float a, b, product;
Scanner scan = new Scanner (System.in);
System.out.print ("Enter any two Numbers: ");
a = scan.nextFloat ();
b = scan.nextFloat ();
product = a * b;
System.out.println ("the product of a and b = " +
product);
}
}
```

The output on the screen:

```
Enter any two Numbers:
```

If you enter two floating point values 2.9 and 3.6

```
the product of a and b = 10.44
```

will be outputted on the screen.

float is used instead of int because a and b are assigned fractional values (i.e., 2.9 and 3.6) if int is used instead of float then the result will not be clearly outputted i.e., instead of 10.44 the computer displays only 10 (as said earlier).

If the statement:

System.out.println ("the product of a and b = " + product);

is replaced by the statement:

System.out.println (a + "* " + b + " = " + product);

Then the output on the screen is:

```
2.9 * 3.6 = 10.44
```

Note: The word public in the statement:

public class HelloWorld

implies: that the program or the data within the program (such as methods, variables etc.) can be accessed directly by an external java program.

If replace the word public by private i.e.,

private class HelloWorld

is written instead of

public class HelloWorld -- then the program or the data within the program (such as methods, variables etc.) cannot be accessed directly by an external program.

If you insert a value 2^3 for a and 3^2 for b, then as said earlier wrong result or compilation error will be flagged on the screen.

```
a=2^3
b=3^2; ---> ERROR
a=2* 2*2
b=3*3; ---> Result will be outputted on the screen i.e.,
```

the product of a and b = 72

If you want to insert a 10 digit number for a and b i.e.,

a=1000000000

b=3000000000, then the statement:

int a, b, product;

should be replaced by the statement:

long int a, b, product;

i.e.,

```
public class HelloWorld{
public static void main (String [] args){
long int a, b, product;
a=1000000000;
b=2000000000;
product = a * b;
System.out.println ("the product of a and b = " + product);
}
}
```

The output on the screen:

the product of a and b = 3000000000000000000

What will be the output of the following program:

```
public class HelloWorld{
static public void main (String args []) {
float x;
x = 2;
System.out.println (" square of a number = " + Math.pow ((x), 2));
}
}
```

Answer:

square of a number = 4

Program 2.2

Java program to find the square of a number

```
public class HelloWorld{
public static void main (String [] args){
int a, b;
a=2;
b = a * a;
System.out.println ("the square of a = " + b);
}
}
```

The output on the screen:

the square of a = 4

If you want to supply the value for a through the key board, then we have to rewrite the above program as follows:

```
import java.util.Scanner;
public class HelloWorld{
public static void main (String [] args) {
int a, b;
Scanner scan = new Scanner (System.in);
System.out.println ("Enter any Number: ");
a = scan.nextInt ();
b = a * a;
System.out.println ("the square of a = " + b);
}
}
```

The output on the screen:

Enter any number:

If you enter a number 3

the square of a = 9 will be outputted on the screen.

Note:

If scan.nextint () is written instead of scan.nextInt ()

public static void main (string [] args); is written instead of public static void main (String [] args)

system.out.println ("the square of a = " + b); is written instead of System.out.println ("the square of a = " + b);

Then the compilation error will be displayed on the screen.

Program 2.3

Java program to find the greatest of two numbers using if - else statement

The syntax of if else statement is:

```
if (this condition is true)
{
print this statement using the println method
}
else
{
print this statement using the println method
}
```

```
public class HelloWorld{
public static void main (String [] args){
int a, b;
a=2;
b =3;
if (a>b)
{
System.out.println ("a is greater than b");
}
else
{
System.out.println ("b is greater than a");
}
}
}
```

The output on the screen:

```
b is greater than a
```

In the above program:

if the condition (a> b) is true, then the statement

```
{
System.out.println ("a is greater than b");
}
```

is executed to print the output:

```
a is greater than b
```

else

the statement

```
{
System.out.println ("b is greater than a");
}
```

is executed to print the output:

```
b is greater than a
```

If you want to supply the values for a and b through the key board, then the above program should be rewritten as:

```
import java.util.Scanner;
public class HelloWorld{
public static void main (String [] args){
int a, b;
Scanner scan = new Scanner (System.in);
System.out.println ("Enter any two Numbers: ");
a = scan.nextInt ();
b = scan.nextInt ();
if (a>b)
{
System.out.println ("a is greater than b");
}
else
{
System.out.println ("b is greater than a");
}
}
}
```

The output on the screen:

```
Enter any two Numbers:
```

If you enter two numbers 2 and 3

```
b is greater than a
```

will be outputted on the screen.

Note:

Even if the statements:

```
System.out.println ("a is greater than b");
System.out.println ("b is greater than a");
```

are not written within the braces { }

i.e.,

```
import java.util.Scanner;
public class HelloWorld{
public static void main (String [] args){
int a, b;
Scanner scan = new Scanner (System.in);
System.out.println ("Enter any two Numbers: ");
a = scan.nextInt ();
b = scan.nextInt ();
if (a>b)
System.out.println ("a is greater than b");
if (b>a)
System.out.println ("b is greater than a");
}
}
```

There will no display of compilation error on the screen or there will be no change in the output displayed on the screen (i.e., b is greater than a will be outputted on the screen).

Program 2.4

Java program to find the greatest of three numbers using if else if else statement

The syntax of if else if else statement is:

```
if (this condition is true)
{
print this statement using the method System.out.println ( );
}
else if (this condition is true)
{
print this statement using the method System.out.println ( );
}
else
{
print this statement using the method System.out.println ( );
}
```

```
public class HelloWorld{
public static void main (String [] args){
int a, b, c;
a=2;
b =3;
c=4;
if (a>b      &    &  a>c)
{
System.out.println ("a is greater than b and c");
}
else if (b>a      &    &  b>c)
{
System.out.println ("b is greater than a and c");
}
else
{
System.out.println ("c is greater than b and a");
}
}
}
```

The output on the screen:

```
c is greater than b and a
```

Note:

If the statements:

```
if (a>b      &    &  a>c)
```

```
{
System.out.println ("a is greater than b and c");
}
else if (b>a & & b>c)
{
System.out.println ("b is greater than a and c");
}
else
{
System.out.println ("c is greater than b and a");
}
```

are replaced by the statements:

```
if (a>b & & a>c)
{
System.out.println (a + "is greater than" + b + "and" + c);
}
else if (b>a & & b>c)
{
System.out.println (b + "is greater than" + a + "and" + c);
}
else
{
System.out.println (c + "is greater than" + b + "and" + a);
}
```

Then the output on the screen is:

```
4 is greater than 3 and 2
```

Program 2.5

Java program to find the average of 10 numbers

```
import java.util.Scanner;
public class HelloWorld{
public static void main (String [] args) {
int N1, N2, N3, N4, N5, N6, N7, N8, N9, N10, X;
Scanner scan = new Scanner (System.in);
System.out.println ("Enter any ten Numbers: ");
N1 = scan.nextInt ();
N2 = scan.nextInt ();
N3 = scan.nextInt ();
N4 = scan.nextInt ();
N5 = scan.nextInt ();
N6 = scan.nextInt ();
N7 = scan.nextInt ();
N8 = scan.nextInt ();
N9 = scan.nextInt ();
N10 = scan.nextInt ();
X = (N1 + N2 + N3 + N4 + N5 + N6 + N7 + N8 + N9 + N10) /10;
System.out.println ("the average of 10 numbers = " + X);
}
}
```

The output on the screen:

```
Enter any ten Numbers:
```

If you enter ten numbers 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10

```
the average of 10 numbers = 5
```

will be outputted on the screen.

Note: The average of 10 numbers is 5.5, the output on the screen is 5 because int is used instead of float.

Program 2.6

Java program to find the simple interest

```
public class HelloWorld{
public static void main (String [] args) {
int P,T, R, SI;
P = 1000;
T = 2;
R = 3;
SI = P*T*R/100;
System.out.println ("the simple interest = " + SI);
}
}
```

The output on the screen:

```
the simple interest = 60
```

If you want to supply the values for P, T and R through the key board, then the above program should take the form:

```
import java.util.Scanner;
public class HelloWorld {
public static void main (String [] args) {
int P,T, R, SI;
Scanner scan = new Scanner (System.in);
System.out.println ("Enter principal amount:");
P = scan.nextInt ();
System.out.println ("Enter time:");
T = scan.nextInt ();
System.out.println ("Enter rate of interest:");
R = scan.nextInt ();
SI = P*T*R/100;
System.out.println ("the simple interest = " + SI);
}
}
```

The output on the screen:

Enter principal amount:

If you enter the principal amount 1000

Enter time:

If you enter the time 2

Enter rate of interest:

If you enter the rate of interest 3

the simple interest = 60

will be outputted on the screen.

Program 2.7

Java program to find the senior citizen

```
public class HelloWorld{
public static void main (String [] args){
int age;
age=20;
if (age> = 60)
{
System.out.println ("senior citizen");
}
else
{
System.out.println ("not a senior citizen");
}
}
}
```

The output on the screen:

```
not a senior citizen
```

(age> = 60) implies age greater than or equal to 60

If you want to supply the value for age through the key board, then the above program should be rewritten as:

```
import java.util.Scanner;
public class HelloWorld{
public static void main (String [] args){
int age;
Scanner scan = new Scanner (System.in);
System.out.println ("Enter the age: ");
age = scan.nextInt ();
if (age> = 60)
{
System.out.println ("senior citizen");
}
else
{
System.out.println ("not a senior citizen");
}
}
}
```

The output on the screen:

```
Enter the age:
```

If you enter the age 60

```
senior citizen
```

will be outputted on the screen.

Suppose if you enter the age 28

```
not a senior citizen
```

will be outputted on the screen.

Program 2.8

Java program to get marks for 3 subjects and declare the result:

If the marks >= 35 in all the subjects the student passes else fails.

```
public class HelloWorld{
public static void main (String [] args){
int M1, M2,M3;
M1 = 38;
M2= 45;
M3 = 67;
if (M1>= 35 & & M2>= 35 & & M3>= 35)
{
System.out.println ("candidate is passed");
}
else
{
System.out.println ("candidate is failed");
}
}
}
```

The output on the screen:

```
candidate is passed
```

(M1>= 35 & & M2>= 35 & & M3>= 35) imply M1 is greater than or equal to 35 and M2 is greater than or equal to 35 and M3 is greater than or equal to 35.

>= imply greater than or equal to.

& & imply and whereas & imply address.

(M1>= 35 & & M2>= 35 & & M3>= 35) is the condition and if the condition:

(M1>= 35 & & M2>= 35 & & M3>= 35) is true, then the statement

```
{
System.out.println ("candidate is passed");
}
```

is executed to print the output:

```
candidate is passed
```

else the statement

```
{
System.out.println ("candidate is failed");
}
```

is executed to print the output:

```
candidate is failed
```

If you want to supply the values for marks M1, M2 and M3 through the key board, then the above program should be rewritten as:

```
import java.util.Scanner;
public class HelloWorld{
public static void main (String [] args) {
int age;
Scanner scan = new Scanner (System.in);
System.out.println ("Enter any three Numbers: ");
M1= scan.nextInt ();
M2 = scan.nextInt ();
M3 = scan.nextInt ();
if (M1>= 35 & & M2>= 35 & & M3>= 35)
{
System.out.println ("candidate is passed");
}
else
{
System.out.println ("candidate is failed");
}
}
}
```

The output on the screen:

```
Enter any three Numbers:
```

If you enter three numbers 26, 28, 39

```
candidate is failed
```

will be outputted on the screen.

Program 2.9

Java program to find profit or loss

```
import java.util.Scanner;
public class HelloWorld{
public static void main (String [] args) {
int CP, SP, loss, profit;
Scanner scan = new Scanner (System.in);
System.out.println ("Enter cost price: ");
CP = scan.nextInt ();
System.out.println ("Enter selling price: ");
SP = scan.nextInt ();
if (SP>CP)
{
System.out.println ("profit= " + (SP-CP));
}
else
{
System.out.println ("loss =" +(CP-SP));
}
}
}
```

The output on the screen:

```
Enter cost price:
```

If you enter the cost price 25

```
Enter selling price:
```

If you enter the selling price 26

```
profit = 1
```

will be outputted on the screen.

If the condition (SP>CP) is true, then the statement:

```
{
System.out.println ("profit= " + (SP-CP));
}
```

is executed to print the output:

profit = (SP-CP) (in this case profit = 26-25 =1)

else the statement:

```
{
System.out.println ("loss =" + (CP-SP));
}
```

is executed to print the output:

loss = (CP-SP)

Program 3.0

Java program to find the incremented and decremented values of two numbers

```
public class HelloWorld{
public static void main (String [] args){
int a, b, c, d, e, f;
a = 10;
b=12;
c=a+1;
d=b+1;
e=a-1;
f=b-1;
System.out.print ("the incremented value of a = "+ c);
System.out.print ("the incremented value of b = "+ d);
System.out.print ("the decremented value of a = "+ e);
System.out.print ("the decremented value of b = "+ f);
}
}
```

The output on the screen:

```
the incremented value of a = 11 the incremented value of b = 13 the decremented value of a = 9 the decremented value of b = 11
```

If the statements:

```
System.out.print ("the incremented value of a = "+ c);
System.out.print ("the incremented value of b = " + d);
System.out.print ("the decremented value of a = " + e);
System.out.print ("the decremented value of b = " + f);
```

are replaced by the statements:

```
System.out.print ("\n the incremented value of a = " + c);
System.out.print ("\n the incremented value of b = " + d);
System.out.print ("\n the decremented value of a = " + e);
System.out.print ("\n the decremented value of b = " + f);
```

i.e., if the above program is rewritten as:

Then the output on the screen is:

```
the incremented value of a = 11
the incremented value of b = 13
the decremented value of a = 9
the decremented value of b = 11
```

i.e., \n make provision for the another result to print in the new line. If you want to supply the values for a and b through the key board, then the above program should take the form:

```
import java.util.Scanner;
public class HelloWorld{
public static void main (String [] args){
int a, b, c, d, e, f;
Scanner scan = new Scanner (System.in);
System.out.println ("Enter any Number: ");
a = scan.nextInt ();
System.out.println ("Enter any Number: ");
b = scan.nextInt ();
c=a+1;
d=b+1;
e=a-1;
f=b-1;
System.out.print ("\n the incremented value of a = " + c);
System.out.print ("\n the incremented value of b = " + d);
System.out.print ("\n the decremented value of a = " + e);
System.out.print ("\n the decremented value of b = " + f);
}
}
```

The output on the screen:

```
Enter any Number:
If you enter the value 2
Enter any Number:
If you enter the value 3
the incremented value of a = 3
the incremented value of b = 4
the decremented value of a = 1
the decremented value of b = 2
```

will be outputted on the screen.

Note: b++ is same as b + 1 and b-- is same as b - 1.

What will be the output of the following programs:

A)

```
import java.util.Scanner;
public class temperature{
public static void main (String [] args) {
float T1, T2, A;
Scanner scan = new Scanner (System.in);
System.out.println ("Enter any Number: ");
T1 = scan.nextFloat ();
System.out.println ("Enter any Number: ");
T2 = scan.nextFloat ();
A = (T1 + T2) / 2;
System.out.println ("the average temperature of the day = " + A);
}
}
```

Answer:

```
Enter any Number:
If you enter the number 2
Enter any Number:
If you enter the number 3
the average temperature of the day = 2.5
```

will be outputted on the screen.

B)

```
import java.util.Scanner;
public class HelloWorld{
public static void main (String [] args) {
int P;
Scanner scan = new Scanner (System.in);
System.out.println ("Enter the percentage: ");
P = scan.nextInt ();
if (P >= 60)
{
System.out.println ("first class");
}
else if (P>=50    &    & P <60)
{
System.out.println ("second class");
}
else
{
System.out.println ("pass class");
}
if (P<40)
{
System.out.println ("fail");
}
}
}
```

Answer:

```
Enter the percentage:
If you enter the number 60
first class
```

will be outputted on the screen.

Program 3.1

Java program to calculate the discounted price and the total price after discount

Given:

If purchase value is greater than 1000, 10% discount

If purchase value is greater than 5000, 20% discount

If purchase value is greater than 10000, 30% discount

discounted price

```
import java.util.Scanner;
public class HelloWorld{
public static void main (String [] args) {
int PV, dis;
Scanner scan = new Scanner (System.in);
System.out.println ("Enter purchased value: ");
PV = scan.nextInt ();
if (PV<1000)
{
System.out.println ("dis = " + PV* 0.1);
}
else if (PV>5000)
{
System.out.println ("dis = " + PV* 0.2);
}
else
{
System.out.println ("dis= " + PV* 0.3);
}
}
}
```

The output on the screen:

```
Enter purchased value:
If you enter the purchased value 6500
dis = 1300
will be outputted on the screen.
```

(PV>1000), (PV>5000) denote the conditions and if the condition (PV>1000) is true i.e., purchased value is greater than 1000, then the statement

```
{
System.out.println ("dis = " + PV* 0.1);
}
```

is executed to print the output:

dis= PV* 10% = PV* 10 /100 = PV* 0.1

and if the condition (PV>1000) is false and if the condition (PV>5000) is true i.e., purchased value is greater than 5000, then the statement

```
{
System.out.println ("dis = " + PV* 0.2);
}
```

is executed to print the output:

dis= PV* 20% = PV* 20 /100 = PV* 0.2

and if the condition (PV>5000) is not true i.e., purchased value is less than 5000, then the statement

```
{
System.out.println ("dis = " + PV* 0.3);
}
```

is executed to print the output:

dis= PV* 30% = PV* 30 /100 = PV* 0.3

total price

```
import java.util.Scanner;
public class HelloWorld{
public static void main (String [] args) {
int PV, total;
Scanner scan = new Scanner (System.in);
System.out.println ("Enter purchased value: ");
PV = scan.nextInt ();
if (PV<1000)
{
System.out.println ("total= " + PV - PV* 0.1);
}
else if (PV>5000)
{
System.out.println ("total = " + PV- PV* 0.2);
}
else
{
System.out.println ("total= " + PV- PV* 0.3);
}
}
}
```

The output on the screen:

```
Enter purchased value:
If you enter the purchased value 650
total = 585
will be outputted on the screen.
```

If the condition (PV>1000) is true i.e., purchased value is greater than 1000, then the statement

```
{
System.out.println ("total= " + PV - PV* 0.1);
}
```

is executed to print the output:

total =PV- dis = PV- PV*10% = PV- PV* 10 /100 = PV - PV * 0.1

and if the condition (PV>1000) is false and if the condition (PV>5000) is true i.e., purchased value is greater than 5000, then the statement

```
{
System.out.println ("total= " + PV - PV* 0.2);
}
```

is executed to print the output:

total =PV- dis = PV- PV*20% = PV- PV* 20 /100 = PV - PV * 0.2

and if the condition (PV> 5000) is not true i.e., purchased value is less than 5000, then the statement

```
{
System.out.println ("total= " + PV - PV* 0.3);
}
```

is executed to print the output:

total =PV- dis = PV- PV*30% = PV- PV* 30 /100 = PV - PV * 0.3

Note: Combing both the programs (above), we can write:

```
import java.util.Scanner;
public class HelloWorld{
public static void main (String [] args){
int PV, dis, total;
Scanner scan = new Scanner (System.in);
System.out.println ("Enter purchased value: ");
PV = scan.nextInt ();
if (PV<1000)
{
System.out.println ("dis = " + PV* 0.1);
System.out.println ("total= " + total - dis);
}
else if (PV>5000)
{
System.out.println ("dis = " + PV* 0.2);
System.out.println ("total= " + total - dis);
}
else
{
System.out.println ("dis = " + PV* 0.3);
System.out.println ("total= " + total - dis);
}
}
}
```

The output on the screen:

```
Enter purchased value:
If you enter the purchased value 850
dis = 85
total = 765
```

will be outputted on the screen.

Program 3.2

Java program to print the first ten natural numbers using for loop statement

```
public class HelloWorld{
public static void main (String [] args){
int i;
for (i=1; i<=10; i++)
System.out.println ("value of i = " + i);
}
}
```

The output on the screen is:

value of i = 1 value of i = 2 value of i= 3 value of i= 4 value of i= 5 value of i= 6 value of i = 7 value of i= 8 value of i = 9 value of i = 10

for (i=1; i<=10; i++) denote the for loop statement and the syntax of the for loop statement is:

for (initialization; condition; increment)

Here:

i=1 denote initialization (i.e., from where to start)

i<=10 denote the condition (i.e., stop when 10 is reached)

i++ imply increment (which tells the value of i to increase by 1 each time the loop is executed) and i++ is the same as i+1.

When a for loop executes, the following occurs:

i = 1

Is the condition (i<=10) is true?

Yes because i=1

The statement System.out.println ("value of i = " + i); is executed to print the output:

value of i = 1

Now, the value of i is:

i = 1+1 = 2

Is the condition (i<=10) is true?

Yes because i=2

The statement System.out.println ("value of i = " + i); is executed to print the output:

value of i = 2

Now, the value of i is:

i = 2+1 = 3

Is the condition (i<=10) is true?

Yes because i=3

The statement System.out.println ("value of i = " + i); is executed to print the output:

value of i = 3

Now, the value of i is:

i = 3+1 = 4

Is the condition (i<=10) is true?

Yes because i=4

The statement System.out.println ("value of i = " + i); is executed to print the output:

value of i = 4

Now, the value of i is:

i = 4+1 = 5

Is the condition (i<=10) is true?

Yes because i=5

The statement System.out.println ("value of i = " + i); is executed to print the output:

value of i = 5

Now, the value of i is:

i = 5+1 = 6

Is the condition (i<=10) is true?

Yes because i=6

The statement System.out.println ("value of i = " + i); is executed to print the output:

value of i = 6

Now, the value of i is:

i = 6+1 = 7

Is the condition (i<=10) is true?

Yes because i=7

The statement System.out.println ("value of i = " + i); is executed to print the output:

value of i = 7

Now, the value of i is:

i = 7+1 = 8

Is the condition (i<=10) is true?

Yes because i=8

The statement System.out.println ("value of i = " + i); is executed to print the output:

value of i = 8

Now, the value of i is:

i = 8+1 = 9

Is the condition (i<=10) is true?

Yes because i=9

The statement System.out.println ("value of i = " + i); is executed to print the output:

value of i = 9

Now, the value of i is:

i = 9+1 = 10

Is the condition (i<=10) is true?

Yes because i=10

The statement System.out.println ("value of i = " + i); is executed to print the output:

value of i = 10

and stop because the condition i<=10 is achieved.

If the statement:

System.out.println ("value of i = " + i);

is replaced by the statement:

System.out.println ("\nvalue of i = " + i);

Then the output on the screen is:

```
value of i = 1
value of i = 2
value of i = 3
value of i = 4
value of i = 5
value of i = 6
value of i = 7
value of i = 8
value of i = 9
value of i = 10
```

If the for loop statement:

for (i=2; i<=10; i++)

is written instead of the statement:

for (i=1; i<=10; i++), then the output on the screen is:

value of i = 2 value of i = 3 value of i= 4 value of i= 5 value of i= 6 value of i = 7 value of i= 8 value of i = 9 value of i= 10

If the for loop statement:

for (i=1; i<10; i++)

is written instead of the statement:

for (i=1; i<=10; i++), then the output on the screen is:

value of i = 1 value of i = 2 value of i= 3 value of i= 4 value of i= 5 value of i= 6 value of i = 7 value of i= 8 value of i = 9

(Note: the condition i<=10 tells to print till value of i =10 but the condition i<10 tells to print till value of i=9)

If the statement:

for (i=1; i=10; i++)

is written instead of the statement:

for (i=1; i<=10; i++), then the output on the screen is:

value of i = 10 value of i = 10 value of i = 10 value of i = 10 value of i= 10 value of i= 10 value of i = 10 value of i= 10 value of i = 10 value of i = 10 value of i = 10 value of i = 10 value of i = 10 value of i = 10 value of i = 10 (continues...).

Note:

If the statement:

System.out.println ("\n value of i = " + i); is replaced by the statement

System.out.println ("\n " + i);

Then the output on the screen is:

```
1
2
3
4
5
6
7
8
9
10
```

What is the mistake in the following program:

```
public class HelloWorld{
public static void main (String []args) throws Exception{
System.out.println ("Hello World");
}
}
```

Answer:

There is no mistake in the above program. Addition of the statement throws Exception does not make any change in the output displayed on the screen or give rise to any compilation error on the screen.

Program 3.3

What will be the output of the following program:

```
public class HelloWorld{
public static void main (String [] args) {
int i;
for (i =1; i<=5; i ++)
System.out.println ("\n Linux is not portable");
}
}
```

Answer:

```
Linux is not portable
Linux is not portable
Linux is not portable
Linux is not portable
Linux is not portable
```

Java program to print the first ten natural numbers using for while loop statement

The syntax of while loop statement is:

```
while (this is the condition)
{
execute this statement;
}
```

```
public class HelloWorld{
public static void main (String [] args)
{
int i = 1;
while (i<=10)
{
System.out.println ("\n " + i++);
}
}
}
```

The output on the screen is:

```
1
2
3
4
5
6
7
8
9
10
```

(i<=10) is the condition and

The statement

System.out.println ("\n " + i++);

is repeatedly executed as long as a given condition (i<=10) is true.

If the statement:

int i=1;

is replaced by the statement:

int i;

Then the compilation error will be displayed on the console screen because initialization is not defined i.e., from where to start is not declared.

If the statement:

int i = 1;

is replaced by the int i = 0;

Then the output on the screen is:

```
0
1
2
3
4
5
6
7
8
9
10
```

Similarly if the statement int i = 0; is replaced by the int i = 7;

Then the output on the screen is:

```
7
8
9
10
```

Java program to print first 10 numbers using do while loop statement

The syntax of do while loop statement is:

```
do
{
execute this statement;
}
while (this is the condition);
```

```
public class HelloWorld{
public static void main (String [] args)
{
int i =1;
do
{
System.out.println (" \n i= " + i++);
} while (i<=10);
}
}
```

The output on the screen is:

```
i=1
i=2
i=3
i=4
i=5
i=6
i=7
i=8
i=9
i=10
```

The statement:

System.out.println (" \n i= " + i++);

is executed and then condition (i<=10) is checked. If condition (i<=10) is true then

The statement:

System.out.println (" \n i= " + i++);

is executed again. This process repeats until the given condition (i<=10) becomes false.

Program 3.4

Java program to print the characters from A to Z using for loop, do while loop and while loop statement.

Java program to print the characters from A to Z using for loop statement:

```
public class HelloWorld{
public static void main (String [] args) {
char a;
for ( a='A'; a<='Z'; a++)
System.out.println ("\n     " + a);
}
}
```

The output on the screen:

```
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
W
X
Y
Z
```

char means the data type is character.

The statement:

char a; imply that we are creating the character a.

If the statement:

for (a=A; a<=Z; a++) is written instead of the statement for (a='A'; a<='Z'; a++)

Then the compilation error will be displayed on the console screen.

Java program to print the characters from A to Z using while loop statement:

```
public class HelloWorld{
public static void main (String [] args) {
char a = 'A';
while (a<='Z')
{
System.out.println ("\n " + a++);
}
}
}
```

Java program to print the characters from A to Z using do while loop statement:

```
public class HelloWorld{
public static void main (String [] args) {
char a = 'A';
do
{
System.out.println ("\n " + a++);
} while (a<='Z');
}
}
```

Program 3.5

Java program to print the given number is even or odd.

```
import java.util.Scanner;
public class HelloWorld{
public static void main (String [] args) {
int a;
Scanner scan = new Scanner (System.in);
System.out.println ("Enter a number: ");
a = scan.nextInt ();
if (a%2 = = 0)
{
System.out.println ("the number is even");
}
else
{
System.out.println ("the number is odd");
}
}
}
```

The output on the screen:

```
Enter a number:
```

If you enter the number 4

```
the number is even
```

will be outputted on the screen.

(a%2 = = 0) is the condition and this condition imply: a divided by 2 yields reminder = 0.

For example: if you enter the number 4

Then a = 4

Then 4 divided by 2 yields the remainder = 0

Then the statement:

```
{
System.out.println ("the number is even");
}
```

is executed to print the output:

```
the number is even
```

(Note: in Java language also = = implies equal to)

If you enter the number 3

Then a = 3

Then 3 divided by 2 yields the remainder = 1

Then the statement

```
{
System.out.println ("the number is odd");
}
```

is executed to print the output:

```
the number is odd
```

Program 3.6

Java program to print the remainder of two numbers

```
import java.util.Scanner;
public class HelloWorld{
public static void main (String [] args) {
int a, b, c;
Scanner scan = new Scanner (System.in);
System.out.println ("Enter a number: ");
a = scan.nextInt ();
System.out.println ("Enter a number: ");
b = scan.nextInt ();
c = a%b;
System.out.println ("the remainder of a and b = " + c);
}
}
```

The output on the screen:

```
Enter a number:
```

If you enter the number 3

```
Enter a number:
```

If you enter the number 2

```
the remainder of a and b = 1
```

will be outputted on the screen.

Since (a =3 and b =2). Therefore:

3 divided by 2 (i.e., a divided by b) yields the remainder equal to 1.

Program 3.7

Java program to check equivalence of two numbers.

```
import java.util.Scanner;
public class HelloWorld{
public static void main (String [] args) {
int x, y;
Scanner scan = new Scanner (System.in);
System.out.println ("Enter a number: ");
x = scan.nextInt ();
System.out.println ("Enter a number: ");
y = scan.nextInt ();
if (x-y==0)
{
System.out.println ("the two numbers are
equivalent");
}
else
{
System.out.println ("the number are not equivalent");
}
}
```

The output on the screen:

Enter a number:

If you enter the number 2

Enter a number:

If you enter the number 2

the two numbers are equivalent

will be outputted on the screen.

Since 2-2 is equal to 0 (i.e., x-y = = 0). Therefore: the statement

```
{
System.out.println ("the two numbers are equivalent");
}
```

is executed to print the output:

two numbers are equivalent

If you enter the integers 3 and 2

The output on the screen is:

the two numbers are not equivalent

Since 3-2 is not equal to 0 (i.e., x-y!= 0). Therefore: the statement

```
{
System.out.println ("the two numbers are not equivalent");
}
```

is executed to print the output:

two numbers are not equivalent

(Note: Like in C & C++, in Java language also!= implies not equal to)

Program 3.8

Java program to print the leap year or not

```
public class HelloWorld{
public static void main (String [] args) {
int year;
year =1996;
if (year%4==0)
{
System.out.println ("leap year");
}
else
{
System.out.println ("not a leap year");
}
}
}
```

The other logic for finding the leap year.

```
import java.util.Scanner;
public class Check_Leap_Year {
public static void main (String args []) {
Scanner s = new Scanner (System.in);
System.out.print ("Enter any year:");
int year = s.nextInt ();
boolean flag = false;
if (year % 400 == 0) {
flag = true;
} else if (year % 100 == 0) {
flag = false;
} else if (year % 4 == 0) {
flag = true;
} else {
flag = false;
}
if (flag) {
System.out.println ("Year "+year+" is a Leap Year");
}
else {
System.out.println ("Year "+year+" is not a Leap Year");
}
}
}
```

The output on the screen:

leap year

Since year =1996. Therefore:

1996 divided by 4 (i.e., year divided by 4) yields the remainder equal to 0.

The statement

```
{
System.out.println ("leap year");
}
```

is executed to print the output:

leap year

If the year is = 1995. Then

1995 divided by 4 (i.e., year divided by 4) yields the remainder not equal to 0.

The statement

```
{
System.out.println ("not a leap year");
}
```

is executed to print the output:

not a leap year

What will be the output on the screen:

```
public class HelloWorld{
int a =5;
public static void main (String [] args){
int a =2;
System.out.println (" value of a = " + a);
}
}
```

Answer:

value of a = 2

If the statement:

System.out.println (" value of a = " + a); is replaced by the statement

System.out.println (" value of a = " +::a); (where:: denote scope resolution operator)

i.e.,

```
public class HelloWorld{
int a =5;
public static void main (String [] args){
int a =2;
System.out.println (" value of a = " +::a);
}
}
```

Then the compilation error will be displayed on the screen because like C++ -- java does not hold / support the resolution operator.

Program 3.9

Java program to print whether the given number is positive or negative

```
public class HelloWorld{
public static void main (String [] args){
int a;
a = -35;
if (a>0)
{
System.out.println ("number is positive");
}
else
{
System.out.println (" number entered is negative");
}
}
}
```

The output on the screen:

```
number entered is negative
```

Since a = -35. Therefore:

a is less than 0 i.e., a<0

The statement

```
{
System.out.println ("number is negative");
}
```

is executed to print the output:

number entered is negative

Program 4.0

Java program to print the sum of the first 10 digits using for loop statement:

```
public class HelloWorld{
public static void main (String [] args) {
int i, sum = 0;
for ( i=1; i<=10; i++)
sum = sum + i;
System.out.println ("sum of the first 10 digits = " + sum);
}
}
```

The output on the screen:

```
sum of the first 10 digits = 55
```

How the sum of the first 10 digits = 55 is outputted on the screen through the for Loop statement?

i=1 (sum = 0 because the sum is initialized to 0 in the statement int i, sum = 0;)

Is i<=10 true?

Yes, do this

sum = sum + i = 0 +1 =1

Now,

i=2 (sum = 1)

Is i<=10 true?

Yes, do this

sum = sum + i = 1 +2 =3

Now,

i=3 (sum = 3)

Is i<=10 true?

Yes, do this

sum = sum + i = 3 +3 = 6

Now,

i=4 (sum = 6)

Is i<=10 true?

Yes, do this

sum = sum + i = 6 + 4= 10

Now,

i=5 (sum = 10)

Is i<=10 true?

Yes, do this

sum = sum + i = 10 + 5= 15

Now,

i=6 (sum = 15)

Is i<=10 true?

Yes, do this

sum = sum + i = 15 + 6 = 21

Now,

i=7 (sum = 21)

Is i<=10 true?

Yes, do this

sum = sum + i = 21 + 7 = 28

Now,

i=8 (sum = 28)

Is i<=10 true?

Yes, do this

sum = sum + i = 28 + 8 = 36

Now,

i=9 (sum = 36)

Is i<=10 true?

Yes, do this

sum = sum + i = 36 + 9 = 45

Now,

i=10 (sum = 45)

Is i<=10 true?

Yes, do this

sum = sum + i = 45 + 10 = 55

stops because the condition i<=10 is achieved

The statement:

System.out.println ("sum of the first 10 digits = " + sum); is executed to display the output:

sum of the first 10 digits = 55
on the screen.
The statement:
System.out.println ("sum of the first 10 digits = " + sum);
is executed to print the output:
sum of the first 10 digits = 55

If the statement:
int i, sum = 0;
is replaced by int i, sum = 1;
Then the output on the screen is:
sum of the first10 digits = 56
What will be the output if the for loop statement for (i =1; i<=10; i++) is replaced by the statement for (i =2; i<10; i++)?
Answer: sum of 10 digits = 44
If the statement int i, sum, sum = 0; is written instead of int i, sum = 0;
Then the compilation error message will be displayed on the screen (stating that sum is twice declared).
If the for loop is ended with a semicolon i.e.,
for (i=1; i<=10; i++);
Then the compilation error will be displayed on the console screen.
Program 4.1
Java program to print the average of the first 10 numbers using for loop statement

```
public class HelloWorld{
public static void main (String [] args){
int i, avg, sum = 0;
for ( i=1; i<=10; i++)
sum = sum + i;
avg = sum/10;
System.out.println ("sum of the first 10 numbers = " + sum);
System.out.println ("average of the first 10 numbers = " + avg);
}
}
```

The output on the screen:

```
sum of the first 10 numbers = 55
average of the first 10 numbers = 5
```

The average of the first 10 numbers = 55/10 = 5.5 not 5. But the output on the screen is:
average of the first 10 numbers = 5
because int is used instead of float.
If the data type float is used i.e.,

```
public class HelloWorld{
public static void main (String [] args) {
float i, avg, sum = 0;
for ( i=1; i<=10; i++)
sum = sum + i;
avg = sum/10;
System.out.println ("sum of the first 10 numbers = " + sum);
System.out.println ("average of the first 10 numbers = " + avg);
}
}
```

The output on the screen:

```
sum of the first 10 numbers = 55
average of the first 10 numbers = 5.5
```

Program 4.2
Java program to print the product of the first 10 digits using for loop statement

```
public class HelloWorld{
public static void main (String [] args) {
int i, product = 1;
for ( i=1; i<=10; i++)
product = product * i;
System.out.println ("the product of the first 10 digits = " + product);
}
}
```

The output on the screen:

```
the product of the first 10 digits = 3628800
```

How the product of the first 10 digits = 3628800 is outputted on the screen through the for Loop statement?
i=1 (product = 1 because the product is initialized to 1 in the statement int i, product = 1;)
Is i<=10 true?
Yes, do this
product = product * i = 1 * 1 =1
Now,
i=2 (product = 1)
Is i<=10 true?
Yes, do this
product = product * i = 1 * 2 = 2
Now,
i=3 (product = 2)
Is i<=10 true?
Yes, do this
product = product * i = 2 * 3 = 6
Now,
i=4 (product = 6)
Is i<=10 true?
Yes, do this
product = product * i = 6 * 4 = 24
Now,
i=5 (product =24)
Is i<=10 true?
Yes, do this
product = product * i = 24 * 5 =120
Now,
i=6 (product =120)
Is i<=10 true?
Yes, do this
product = product * i = 120 * 6 = 720

Now,
i=7 (product =720)
Is i<=10 true?
Yes, do this
product = product * i = 720 * 7 = 5040
Now,
i=8 (product =5040)
Is i<=10 true?
Yes, do this
product = product * i = 5040 * 8 = 40320
Now,
i=9 (product = 40320)
Is i<=10 true?
Yes, do this
product = product * i = 40320 * 9 = 362880
Now,
i=10 (product = 362880)
Is i<=10 true?
Yes, do this
product = product * i = 362880 * 10 = 3628800
stops because the condition i<=10 is achieved.
The statement:
System.out.println ("the product of the first10 digits = " + product); is executed to display the output:
the product of the first 10 digits = 3628800
If the statement int i, product = 1; is replaced by int i, product = 0;
Then the output on the screen is:
the product of the first 10 digits = 0
If the statement for (i=1; i<=10; i++) is replaced by for (i=5; i<=8; i++)
Then the output on the screen is:
the product of the first 10 digits = 1680

Program 4.3
Java Program to print the table of a number using the for loop statement

```
import java.util.Scanner;
public class HelloWorld{
public static void main (String [] args){
int n, i;
Scanner scan = new Scanner (System.in);
System.out.println ("Enter a number: ");
n = scan.nextInt ();
for ( i=1; i<=5; i++)
System.out.println ( \n n + " *  " + i + " = " + n
* i);
}
}
```

Output on the screen:

```
Enter any number:
If you enter the number 2 (i.e., n=2)
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
```

will be outputted on the screen.
How the execution takes its Way through the for Loop statement
Since you entered the number 2, therefore: n=2.
i=1
Is i<=5 true?
Yes, print this
2 * 1 = 2
using the statement System.out.println (\n n + " * " + i + " = " + n * i);
Now,
i=2
Is i<=5 true?
Yes, print this
2 * 2 = 4
using the statement System.out.println (\n n + " * " + i + " = " + n * i);
Now,
i=3
Is i<=5 true?
Yes, print this
2 * 3 = 6
using the statement System.out.println (\n n + " * " + i + " = " + n * i);
Now,
i=4
Is i<=5 true?
Yes, print this
2 * 4 = 8
using the statement System.out.println (\n n + " * " + i + " = " + n * i);
Now,
i=5
Is i<=5 true?
Yes, print this
2 * 5 = 10
using the statement System.out.println (\n n + " * " + i + " = " + n * i);
stop Now because the condition i <=5 is achieved.
If the symbol * is replaced by +
i.e.,

```
import java.util.Scanner;
public class HelloWorld{
public static void main (String [] args){
int n, i;
Scanner scan = new Scanner (System.in);
System.out.println ("Enter a number: ");
n = scan.nextInt ();
for ( i=1; i<=5; i++)
System.out.println ( \n n + " +  " + i + " = " + n
+ i);
}
}
```

Then the output on the screen is:

```
Enter any number:
If you enter the number 2 (i.e., n=2)
2 + 1 = 3
2 + 2 = 4
2 + 3 = 5
2 + 4 = 6
2 + 5 = 7
```

will be outputted on the screen.

Program 4.4

Java program to print the first 10 numbers starting from one together with their squares

```
public class HelloWorld{
public static void main (String [] args){
int i;
for ( i=1; i<=10; i++)
System.out.println (" number = " + i + " its square = " + i*i);
}
}
```

The output on the screen:

```
number = 1 its square=1number = 2 its square=4number = 3 its square=9number = 4 its square=16number = 5 its square=25number = 6 its square=36number = 7 its square=49number = 8 its square=64number = 9 its square=81number= 10 its square=100
```

If the statement:

System.out.println (" number = " + i + " its square = " + i*i); is replaced by the statement

System.out.println (" \n number = " + i + " its square = " + i*i);

i.e., if the above program is rewritten as:

```
public class HelloWorld{
public static void main (String [] args){
int i;
for ( i=1; i<=10; i++)
System.out.println (" \n number = " + i + " its square = " + i*i);
}
}
```

Then the output on the screen is:

```
number = 1 its square=1
number = 2 its square=4
number = 3 its square=9
number = 4 its square=16
number = 5 its square=25
number = 6 its square=36
number = 7 its square=49
number = 8 its square=64
number = 9 its square=81
number= 10 its square=100
```

If the statement:

System.out.println (" \n number = " + i + " its square = " + i*i); is replaced by the statement:

System.out.println (" \n number = " + i + " \t its square = " + i*i);

i.e., if the above program is rewritten as:

```
public class HelloWorld{
public static void main (String [] args){
int i;
for ( i=1; i<=10; i++)
System.out.println (" \n number = " + i + " \t its square = " + i*i);
}
}
```

Then the output on the screen is:

```
number=1      its square=1
number=2      its square=4
number=3      its square=9
number=4      its square=16
number=5      its square=25
number=6      its square=36
number=7      its square=49
number=8      its square=64
number=9      its square=81
number=10     its square=100
```

tab /t is included because to leave space between

number =1 and its square=1

If the statement:

System.out.println (" \n number = " + i + " \t its square = " + i*i); is replaced by the statement:

System.out.println (" \n number = " + i + " \n its square = " + i*i);

i.e., if the above program is rewritten as:

```
public class HelloWorld{
public static void main (String [] args){
int i;
for ( i=1; i<=10; i++)
System.out.println (" \n number = " + i + " \n its square = " + i*i);
}
}
```

Then the output on the screen is:

```
number = 1
its square=1
number = 2
its square=4
number = 3
its square=9
number = 4
its square=16
number = 5
its square=25
number = 6
its square=36
number = 7
its square=49
number = 8
its square=64
number = 9
its square=81
```

```
number= 10
its square=100
```

Write a program to print the first 10 numbers starting from one together with their squares and cubes:

Answer:

```
public class HelloWorld{
public static void main (String [] args) throws Exception{
int i;
for ( i=1; i<=10; i++)
System.out.println (" \n number = " + i + " its square = " + i*i + " its cube = " + i*i*i);
}
}
```

Program 4.5

Java program to print the sum of two numbers using method

```
public class HelloWorld{
public static void main (String [] args){
int a, b, c;
a = 11;
b = 6;
c = add (a, b);
System.out.println (" sum of two numbers = " + c);
}
public static int add (int a, int b) {
return (a+b);
}
}
```

The output on the screen:

```
sum of two numbers = 17
```

There are 2 methods in the above program:

public static void main (String [] args)

public static int add (int a, int b)

public static void main (String [] args) imply: main method and

{

} imply the body of the main method with in which the program statements:

```
int a, b, c;
a = 11;
b = 6;
c = add (a, b);
System.out.println (" sum of two numbers = " + c);
```

are written.

Like in C ++ (the function declaration is not made) and unlike in C ((the function declaration is made) -- there is no need for method declaration in Java (i.e., without the method declaration the program will be successfully executed and the result will be outputted on the screen)

public static int add (int a, int b) imply: the method to add two integers x and y and

```
{
return (a+b);
}
```

} imply the body of the method public static int add (int a, int b)

main method:

public static void main (String [] args)

and the method:

public static int add (int a, int b)

should be written inside the body of the public class HelloWorld.

The statement int a, b, c; imply that we creating the integer variables a, b and c.

The statements:

```
a = 11;
b = 6;
c = add (a, b);
```

imply that we are assigning the values to the created variables.

The statement:

c = add (x, y); imply method call (i.e., we are calling the method public static int add (int a, int b) to add the values (i.e., 11 and 6) and return the result (i.e., 17) to the statement System.out.println (" sum of two numbers = " + c); to make provision to display the output of the sum of two entered numbers as 17 on the screen.

Java program to print the product of two numbers using method

```
public class HelloWorld{
public static void main (String [] args) {
int a, b, c;
a = 2;
b = 3;
c = mult (a, b);
System.out.println (" product of two numbers = " + c);
}
public static int mult (int a, int b){
return (a*b);
}
}
```

The output on the screen:

```
product of two numbers = 6
```

will be outputted on the screen.

Java program to print the greatest of two numbers using method

```
import java.util.Scanner;
public class HelloWorld{
public static void main (String [] args) {
int a, b;
Scanner scan = new Scanner (System.in);
System.out.println ("Enter any two numbers: ");
a = scan.nextInt ();
b = scan.nextInt ();
System.out.println (" largest of two numbers = " + max (a, b) );
}
public static int max (int a, int b) {
if (a>b)
return a;
else
return b;
}
}
```

The output on the screen:

```
Enter any two numbers:
```

If you enter two numbers 5 and 2

```
largest of two numbers= 5
```

will be outputted on the screen.

Java program to print the greatest of three numbers using method

```
import java.util.Scanner;
public class HelloWorld{
public static void main (String [] args) {
int a, b, c;
Scanner scan = new Scanner (System.in);
System.out.println ("Enter any three numbers: ");
a = scan.nextInt ();
b = scan.nextInt ();
c= scan.nextInt ();
System.out.println (" largest of two numbers = " + max (a, b, c) );
}
public static int max (int a, int b, int c) {
if (a>b   &   &   a>c)
return a;
else if (b>c   &   &   b>a)
return b;
else
return c;
}
}
```

The output on the screen:

```
Enter any three numbers:
```

If you enter three numbers 3, 5 and 10

```
largest of three numbers = 10
```

will be outputted on the screen.

Java program to print the square of the number using method

```
import java.util.Scanner;
public class HelloWorld{
public static void main (String [] args) {
int x;
Scanner scan = new Scanner (System.in);
System.out.println ("Enter any number: ");
x = scan.nextInt ();
System.out.println ("square of the number = " + square (x));
}
public static int square (int x){
return x*x;
}
}
```

The output on the screen is:

```
Enter any number:
```

If you enter the number 5

```
square of the number = 25
```

will be outputted on the screen.

Program 4.6

Switch (case) allows to make decision from the number of choices i.e., from the number of cases

For example:

```
public class HelloWorld{
public static void main (String [] args)throws Exception{
char ch;
System.out.print ("Enter a character:");
ch = (char)System.in.read ();
switch (ch)
{
case 'R':
System.out.print ("Red");
break;
case 'W':
System.out.print ("White");
break;
case 'Y':
System.out.print ("Yellow");
break;
case 'G':
System.out.print ("Green");
break;
default:
System.out.print ("Error");
break;
}
}
}
```

The output on the screen is:

```
Enter a character:
```

If you enter a character R

```
Red
```

will be outputted on the screen.

switch (ch) allow to make decision from the number of choices i.e., from the number of cases

```
case 'R':
case 'W':
case 'Y':
case 'G':
```

Since we have entered the character R (which corresponds to case 'R':)

The statement:

```
System.out.print ("Red");
```

is executed to display the output:

```
Red
```

on the screen.

Suppose you enter a character K

The output on the screen is:
Error
(Entered character K does not correspond to any of the cases
case 'R':
case 'W':
case 'Y':
case 'G':
Therefore the statements:
default:
System.out.print ("Error");
are executed to display the output:
Error
on the screen).
If the statements:

```
{
case 'R':
System.out.print ("Red");
break;
case 'W':
System.out.print ("White");
break;
case 'Y':
System.out.print ("Yellow");
break;
case 'G':
System.out.print ("Green");
break;
default:
System.out.print ("Error");
break;
}
```

are replaced by the statements:

```
{
case 'R':
System.out.print ("Red");
case 'W':
System.out.print ("White");
case 'Y':
System.out.print ("Yellow");
break;
case 'G':
System.out.print ("Green");
break;
default:
System.out.print ("Error");
break;
}
```

Then the output on the screen is:

```
Red
White
Yellow
```

i.e., the output is printed till yellow even though you have entered the character R.

Note: C and C++ supports pointers and structures whereas Java does not i.e., Java do not support structures and pointers because JVM (Java virtual machine a core component of java) do not support structures and pointers.

Program 4.7
Java program to print the output

```
Element [0] = 16
Element [1] = 18
Element [2] = 20
Element [3] = 25
Element [4] = 36
```

using arrays:

```
public class HelloWorld{
public static void main (String [] args){
int i;
int [] num = {16, 18, 20, 25, 36};
for (i=0; i<5; i++)
System.out.println ("Element [" + i + " ] = " + num [i]);
}
}
```

The output on the screen:

```
Element [0] = 16
Element [1] = 18
Element [2] = 20
Element [3] = 25
Element [4] = 36
```

Ends because of the condition i<5.
Note:
//--//-
Array declaration in C:
int num [5] = {16, 18, 20, 25, 36};
or
int num [] = {16, 18, 20, 25, 36};
Array declaration in C++:
int num [5] = {16, 18, 20, 25, 36};
or
int num [] = {16, 18, 20, 25, 36};
But array declaration in java:
int [] num = {16, 18, 20, 25, 36};
//---//

The statement:
int [] num = {16, 18, 20, 25, 36}; imply that we are creating an integer array (and the name of array is num) consisting of 5 values (i.e., 16, 18, 20, 25, 36) of the same data type int.

With the declaration int [] num = {16, 18, 20, 25, 36}; -- computer creates 5 memory cells (because there are 5 elements within the braces {}) with name num [0], num [1], num [2], num [3], num [4]. And since
int [ ] num = {16, 18, 20, 25, 36};
the values 16, 18, 20, 25, 36 are stored in num [0], num [1], num [2], num [3], num [4] respectively.

How the execution takes its way through the for loop statement

i=0

Is i<5 true?

Yes, print this

Element [0] = 16

using the statement:

System.out.println ("Element [" + i + "] = " + num [i]);

Now,

i=1

Is i<5 true?

Yes, print this

Element [1] = 18

using the statement:

System.out.println ("Element [" + i + "] = " + num [i]);

Now,

i=2

Is i<5 true?

Yes, print this

Element [2] = 20

using the statement:

System.out.println ("Element [" + i + "] = " + num [i]);

Now,

i=3

Is i<5 true?

Yes, print this

Element [3] = 25

using the statement:

System.out.println ("Element [" + i + "] = " + num [i]);

Now,

i=4

Is i<5 true?

Yes, print this

Element [4] = 36

using the statement:

System.out.println ("Element [" + i + "] = " + num [i]);

Stop because the condition is i<5.

If i<=5 i.e., if the for loop statement was

for (i=0; i<=5; i++)

Then the output on the screen is:

```
Element [0] = 16
Element [1] = 18
Element [2] = 20
Element [3] = 25
Element [4] = 36
Element [5] = 365
```

365 is the number stored in the memory i.e., any number stored in the memory will be displayed.

If the statement int [] num = {16, 18, 20, 25, 36}; is replaced by the statement:

int [5] num = {16, 18, 20, 25, 36};

or by the statement:

int num [i] = {16, 18, 20, 25, 36};

Then the compilation error will be displayed on the screen.

Suppose the statement:

System.out.println ("Element [" + i + "] = " + num [i]); is replaced by the statement:

System.out.println ("Element [" + i + "] = " + num [0]);

Then the output on the screen is:

```
Element [0] = 16
Element [1] = 16
Element [2] = 16
Element [3] = 16
Element [4] = 16
```

Suppose the statement:

System.out.println ("Element [" + i + "] = " + num [i]); is replaced by the statement:

System.out.println ("Element [" + i + "] = " + num [1]);

Then the output on the screen is:

```
Element [0] = 18
Element [1] = 18
Element [2] = 18
Element [3] = 18
Element [4] = 18
```

Suppose the statement:

System.out.println ("Element [" + i + "] = " + num [i]); is replaced by the statement:

System.out.println ("Element [" + i + "] = " + num [2]);

Then the output on the screen is:

```
Element [0] = 20
Element [1] = 20
Element [2] = 20
Element [3] = 20
Element [4] = 20
```

Suppose the statement:

System.out.println ("Element [" + i + "] = " + num [i]); is replaced by the statement:

System.out.println ("Element [" + i + "] = " + num [3]);

Then the output on the screen is:

```
Element [0] = 25
Element [1] = 25
Element [2] = 25
Element [3] = 25
Element [4] = 25
```

Suppose the statement System.out.println ("Element [" + i + "] = " + num [i]); is replaced by the statement

System.out.println ("Element [" + i + "] = " + num [4]);

Then the output on the screen is:

```
Element [0] = 36
Element [1] = 36
Element [2] = 36
```

Element [3] = 36
Element [4] = 36

Java program to print the sum of the elements in array.

```
public class HelloWorld{
public static void main (String [] args){
int i, sum = 0;
int [] num = {16, 18, 20, 25, 36};
for (i=0; i<5; i++)
sum = sum + num [i];
System.out.println ("Sum of the Elements in the array = " + sum);
}
}
```

The output on the screen:

```
Sum of the Elements in the array = 115
```

i.e., 16 + 18 + 20 + 25 + 36 = 115

How the Execution takes its way through the for loop statement

i=0 (sum = 0)
Is i<5 true?
Yes, do this
sum = sum + num [i] = sum + num [0] = 0 +16 =16
Now,
i=1 (sum = 16)
Is i<5 true?
Yes, do this
sum = sum + num [i] = sum + num [1] = 16 +18 =34
Now,
i=2 (sum = 34)
Is i<5 true?
Yes, do this
sum = sum + num [i] = sum + num [2] = 34 +20 =54
Now,
i=3 (sum = 54)
Is i<5 true?
Yes, do this
sum = sum + num [i] = sum + num [3] = 54 +25 =79
Now,
i=5 (sum = 79)
Is i<5 true?
Yes, do this
sum = sum + num [i] = sum + num [5] = 79 + 36 =115
stop because the condition i<5 is achieved

The statement:
System.out.println ("Sum of the Elements in the array = " + sum); is executed to display the output:
Sum of the Elements in the array = 115
on the screen.

If the statement:
int i, sum = 0;
is replaced by int i, sum = 1;
Then The output on the screen:
Sum of the Elements in the array = 116

Java program to print the average of the elements in the array

```
public class HelloWorld{
public static void main (String [] args){
int i, avg, sum = 0;
int [] num = {16, 18, 20, 25, 36};
for (i=0; i<5; i++)
sum = sum + num [i];
avg = sum/5;
System.out.println ("Sum of the Elements in the array = " + sum);
System.out.println ("average of the Elements in the array = " + avg);
}
}
```

The output on the screen:

```
Sum of the Elements in the array = 115
average of the elements in the array = 23
```

Write a program to print
Einstein [0] = E
Einstein [1] = I
Einstein [2] = N
Einstein [3] = S
Einstein [4] = T
Einstein [5] = E
Einstein [6] = I
Einstein [7] = N
using arrays

Answer:

```
public class HelloWorld{
public static void main (String [] args) throws Exception{
int i;
char [] num = {'E', 'I', 'N', 'S', 'T', 'E', 'I', 'N'};
for (i=0; i<8; i++)
System.out.println ("Einstein [" + i + " ] = " + num [i]);
}
}
```

What will be the output of the following programs?

i)

```
public class HelloWorld{
public static void main (String [] args) throws Exception{
int i;
int [] name = {'E', 'I', 'N', 'S', 'T ', 'E', 'I', 'N'};
for (i=0; i<8; i++)
System.out.println ("Einstein [" + i + " ] = " + name [i]);
}
}
```

Answer:

```
Einstein [0] = 69
Einstein [1] = 73
Einstein [2] = 78
Einstein [3] = 83
Einstein [4] = 84
Einstein [5] = 69
Einstein [6] = 73
Einstein [7] = 78
```

ii)

```
public class HelloWorld{
public static void main (String [] args) throws Exception{
int i;
char [] body = {'b', 'o', 'd', 'y'};
for (i=0; i<4; i++)
System.out.println ("body [" + body [i] + " ] = " + body [i]);
}
}
```

Answer:

```
body [b] = b
body [o] = o
body [d] = d
body [y] = y
```

Note:

//---

```
import java.util.Scanner;
public class HelloWorld {
public static void main (String [] args) {
int x, y;
Scanner scan = new Scanner (System.in);
System.out.print ("Enter any Number: ");
x = scan.nextFloat ();
System.out.print ("Enter any Number: ");
y = scan.nextInt ();
System.out.println (" square root of x = " + Math.sqrt (x));
System.out.println (" square root of y = " + Math.sqrt (y));
}
}
```

The output on the screen:

```
Enter any Number:
```

If you enter the number 9

```
square root of x = 3
```

will be outputted on the screen.

```
Enter any Number:
```

If you enter the number 4

```
square root of y = 2
```

will be outputted on the screen.

If

/*

*/

is introduced i.e., if the above program is rewritten as:

```
import java.util.Scanner;
public class HelloWorld {
public static void main (String [] args) {
int x, y;
Scanner scan = new Scanner (System.in);
System.out.print ("Enter any Number: ");
x = scan.nextInt ();
/*
System.out.print ("Enter any Number: ");
y = scan.nextInt ();
*/
System.out.println (" square root of x = " + Math.sqrt (x));
/*
System.out.println (" square root of y = " + Math.sqrt (y));
*/
}
}
```

Then the output on the screen is:

```
Enter any Number:
```

If you enter the number 9

```
square root of x = 3
```

will be outputted on the screen.

---//

What is the mistake in the following program:

```
public class HelloWorld {
public static void main (String [] args) {
long float x;
Scanner scan = new Scanner (System.in);
System.out.print ("Enter any Number: ");
x = scan.nextFloat ();
System.out.println (" square root of x = " + Math.cbrt (x));
}
```

Answer:

long float x; should not be used -- only float x should be used because Java do not support the data type such as long int, long float etc.

Program 4.8

continue and break statements:

A)

```
public class HelloWorld{
public static void main (String []args){
int i;
for (i=1; i<=5; i++){
if (i==3){
continue;
}
System.out.println ("" + i);
}
}
}
```

Output on the screen:

```
1
2
4
5
```

B)

```
public class HelloWorld {
public static void main (String []args){
int i;
for (i=1; i<=5; i++){
if (i==3){
break;
}
System.out.println ("" + i);
}
}
}
```

Output on the screen:

```
1
2
```

What will be the output of the following program:

```
public class HelloWorld {
public static void main (String args []){
System.out.println (Math.max (1269, 1356));
}
}
```

Output on the screen:

```
1356
```

Note:

//--

Abstraction hiding implementation details from the user by providing interface

Encapsulation hiding data

In the statement:

"1 + 2"

"1" and "2" imply the operands and the plus symbol imply the operator.

Polymorphism

Suppose if you are in class room that time you behave like a student, when you are in shopping mall at that time you behave like a customer, when you at your home at that time you behave like a son or daughter. Your ability to present in different-different behaviors is known as polymorphism.

In the example:

```
public class HelloWorld
{
public static void main (String [] args)
{
int a, b, sum;
a=1;
b=2;
sum = a + b;
System.out.println ("the sum of a and b = " + sum);
}
}
```

Plus symbol ("+") act as an arithmetic operator in the statement:

sum = a+b;

and it act as the concatenation operator in the statement:

System.out.println ("the sum of a and b = " + sum);

The ability of plus symbol to behave both as arithmetic operator and concatenation operator is known as polymorphism.

Inheritance

```
public class game {
}
public class player extends game{
}
```

Here public class player extends game implies:

class player is public and it is the sub class of the class game.

Since class player is the subclass of class game -- class player automatically takes on all the behavior and attributes of its parent class "game" i.e., methods or fields within the class game will be automatically be included in the class player.

Note:

The statements:

public class player extends game

public class game extends ball

implies: that class player is not only a subclass of class game but also it is a subclass of class ball.

Encapsulation

```
public class Account {
        private decimal accountBalance = 500.00;
        public decimal CheckBalance () {
            return accountBalance;
        }
    }
    /* accountBalance can be checked via public "CheckBalance" method provided by the "Account" class
but its value cannot be manipulated because data variable accountBalance is declared private */
```

Encapsulation is the technique of bringing the data variables and methods in single frame and declaring data variable private (so it cannot be accessed by anyone outside the class, thereby hiding / encapsulating the data variable (String name) within the public class Student) and providing indirect access to the data variable via public methods.

---------------//

Comparison of C, C++ and Java

C & C++ support pointers and structures while Java do not.

The code of C and C++ are directly converted into machine level language and it is executed while the code of Java is converted into Java byte codes and then it is converted into machine level language and it is executed.

C uses scanf as input function to read the character or integer entered through the keyboard and printf as output function to print the output on the screen.

C++ uses cin as input function to read the character or integer entered through the keyboard and cout as output function to print the output on the screen.

But Java uses scan.nextInt () or scan.nextFloat () as input method to read the variable entered through the keyboard and System.out.println as output method to print the output on the screen.

Functions are in C & C++ whereas methods are in Java.

C & C++ are platform dependent whereas Java is platform independent (Code written in Java can be taken from one computer to the other without having to worry about system configuration details).

In C & C++, program instruction codes are written and executed within the body of main function main () where as in Java -- program instruction codes are written and executed within the body of main method public static void main (String [] args).

data types like int float, char are same in C, C++ & Java.

C is structured language whereas C++ &

Java is object oriented language (i.e., C++ & Java has the extensive power and immense extensibility to write large scale complex programs).

Operators such as %d, %f & %c are used in C whereas no operators are used in C++ & Java.

A program written in Java usually requires more memory space than the same program written in C & C++ and it is fast, reliable, and secure. According to Oracle, the company that owns Java, Java runs on 3 billion devices worldwide.

Java provides both high speed and high performance and reliability, flexibility and seamless integration with other frameworks and technologies -- compared to C & C++.

Java is a popular general-purpose programming language and computing platform that supports multithreading (a process of executing several codes concurrently) while C & C++ do not.

One of the reasons why Java is widely used is because of the availability of huge standard library that consists hundreds of classes and methods under different packages to help software developers.

For example:

java.lang for advanced features of strings, arrays etc.

java.util for data structures, regular expressions, date and time functions etc.

java.io for file i/o, exception handling etc.

The object oriented programming in Java

The Java programming came in the midst of several programming languages which had object oriented features in their arsenal. Their were programming languages like smalltalk and C++ which were handling the object oriented programming. But, the idea was make programming easier, which could only be done by mounting up the piles of libraries of classes and function which could be used to solve complex programming problems.

The idea behind building a programming language was that every tool in the language has to be an object. As objects can be reliably used as different instances of the program adding up to the reuse of code and portability.

The Object class is the base class of all the classes in Java. Lets go through some example of exception handling and files before diving into oriented programming in Java.

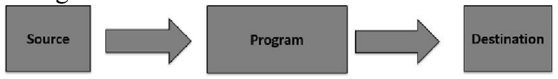
The java.io package contains nearly every class you might ever need to perform input and output (I/O) in Java. All these streams represent an input source and an output destination. The stream in the java.io package supports many data such as primitives, object, localized characters, etc.

Stream

A stream can be defined as a sequence of data. There are two kinds of Streams

InPutStream The InputStream is used to read data from a source.

OutPutStream The OutputStream is used for writing data to a destination.



Java provides strong but flexible support for I/O related to files and networks but this tutorial covers very basic functionality related to streams and I/O. We will see the most commonly used examples one by one

Byte Streams

Java byte streams are used to perform input and output of 8-bit bytes. Though there are many classes related to byte streams but the most frequently used classes are, FileInputStream and FileOutputStream. Following is an example which makes use of these two classes to copy an input file into an output file

```
import java.io.*; // This library is required to execute I/O operations on files.
publicclassCopyFile{
publicstaticvoid main (String args [])throwsIOException{
FileInputStreamin=null;
FileOutputStreamout=null;
try{
in=newFileInputStream ("input.txt");
out=newFileOutputStream ("output.txt");
int c;
while ((c =in.read ())!=-1){
out.write (c);
}
}finally{
if (in!=null){
in.close ();
}
if (out!=null){
out.close ();
}
}
}
}
```

Now let's have a file input.txtwith the following content

This is test for copy file.

As a next step, compile the above program and execute it, which will result in creating output.txt file with the same content as we have in input.txt. So let's put the above code in CopyFile.java file and do the following

```
$javac CopyFile.java
$java CopyFile
```

Character Streams

JavaBytestreams are used to perform input and output of 8-bit bytes, whereas JavaCharacterstreams are used to perform input and output for 16-bit unicode. Though there are many classes related to character streams but the most frequently used classes are, FileReaderand FileWriter. Though internally FileReader uses FileInputStream and FileWriter uses FileOutputStream but here the major difference is that FileReader reads two bytes at a time and FileWriter writes two bytes at a time.

We can re-write the above example, which makes the use of these two classes to copy an input file (having unicode characters) into an output file

Example

```
import java.io.*;
publicclassCopyFile{
publicstaticvoid main (String args [])throwsIOException{
FileReaderin=null;
FileWriterout=null;
try{
in=newFileReader ("input.txt");
out=newFileWriter ("output.txt");
int c;
while ((c =in.read ())!=-1){
out.write (c);
}
}finally{
if (in!=null){
in.close ();
}
if (out!=null){
out.close ();
}
}
}
}
```

Now let's have a file input.txtwith the following content

This is test for copy file.

As a next step, compile the above program and execute it, which will result in creating output.txt file with the same content as we have in input.txt. So let's put the above code in CopyFile.java file and do the following

```
$javac CopyFile.java
$java CopyFile
```

Standard Streams

All the programming languages provide support for standard I/O where the user's program can take input from a keyboard and then produce an output on the computer screen. If you are aware of C or C++ programming languages, then you must be aware of three standard devices STDIN, STDOUT and STDERR. Similarly, Java provides the following three standard streams

Standard Input This is used to feed the data to user's program and usually a keyboard is used as standard input stream and represented as System.in.

Standard Output This is used to output the data produced by the user's program and usually a computer screen is used for standard output stream and represented as System.out.

Standard Error This is used to output the error data produced by the user's program and usually a computer screen is used for standard error stream and represented as System.err.

Following is a simple program, which creates InputStreamReaderto read standard input stream until the user types a "q"

Example

```
import java.io.*;
publicclassReadConsole{
publicstaticvoid main (String args []throwsIOException{
InputStreamReader cin =null;
try{
cin =newInputStreamReader (System.in);
System.out.println ("Enter characters, 'q' to quit.");
char c;
do{
c =(char) cin.read ();
System.out.print (c);
}while (c!='q');
}finally{
if (cin!=null){
cin.close ();
}
}
}
}
```

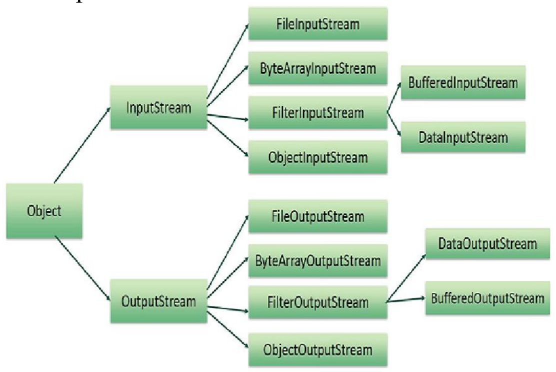
Let's keep the above code in ReadConsole.java file and try to compile and execute it as shown in the following program. This program continues to read and output the same character until we press 'q'

```
$javac ReadConsole.java
$java ReadConsole
Enter characters, 'q' to quit.
1
1
e
e
q
q
```

Reading and Writing Files

As described earlier, a stream can be defined as a sequence of data. The InputStream is used to read data from a source and the OutputStream is used for writing data to a destination.

Here is a hierarchy of classes to deal with Input and Output streams.



The two important streams areFileInputStreamandFileOutputStream, which would be discussed in this tutorial.

FileInputStream

This stream is used for reading data from the files. Objects can be created using the keywordnewand there are several types of constructors available.

Following constructor takes a file name as a string to create an input stream object to read the file

InputStream f = new FileInputStream ("C:/java/hello");

Following constructor takes a file object to create an input stream object to read the file. First we create a file object using File () method as follows

File f = new File ("C:/java/hello");

InputStream f = new FileInputStream (f);

Once you haveInputStreamobject in hand, then there is a list of helper methods which can be used to read to stream or to do other operations on the stream.

There are other important input streams available, for more detail you can refer to the following links

ByteArrayInputStream

DataInputStream

FileOutputStream

FileOutputStream is used to create a file and write data into it. The stream would create a file, if it doesn't already exist, before opening it for output.

Here are two constructors which can be used to create a FileOutputStream object.

Following constructor takes a file name as a string to create an input stream object to write the file

OutputStream f = new FileOutputStream ("C:/java/hello")

Following constructor takes a file object to create an output stream object to write the file. First, we create a file object using File () method as follows

File f = new File ("C:/java/hello");

OutputStream f = new FileOutputStream (f);

Sr.No.	Method & Description
1	public void close () throws IOException{} This method closes the file output stream. Releases any system resources associated with the file. Throws an IOException.
2	protected void finalize ()throws IOException {} This method cleans up the connection to the file. Ensures that the close method of this file output stream is called when there are no more references to this stream. Throws an IOException.
3	public int read (int r)throws IOException{} This method reads the specified byte of data from the InputStream. Returns an int. Returns the next byte of data and -1 will be returned if it's the end of the file.
4	public int read (byte [] r) throws IOException{} This method reads r.length bytes from the input stream into an array. Returns the total number of bytes read. If it is the end of the file, -1 will be returned.
5	public int available () throws IOException{} Gives the number of bytes that can be read from this file input stream. Returns an int.

Once you haveOutputStreamobject in hand, then there is a list of helper methods, which can be used to write to stream or to do other operations on the stream.

Sr.No.	Method & Description
1	public void close () throws IOException{} This method closes the file output stream. Releases any system resources associated with the file. Throws an IOException.
2	protected void finalize ()throws IOException {} This method cleans up the connection to the file. Ensures that the close method of this file output stream is called when there are no more references to this stream. Throws an IOException.
3	public void write (int w)throws IOException{} This methods writes the specified byte to the output stream.
4	public void write (byte [] w) Writes w.length bytes from the mentioned byte array to the OutputStream.

There are other important output streams available, for more detail you can refer to the following links

ByteArrayOutputStream

DataOutputStream

Example

Following is the example to demonstrate InputStream and OutputStream

```
import java.io.*;
publicclass fileStreamTest {
publicstaticvoid main (String args []){
try{
byte bWrite []={11,21,3,40,5};
OutputStream os =newFileOutputStream
("test.txt");
for (int x =0; x < bWrite.length; x++){
os.write ( bWrite [x]);// writes the bytes
}
os.close ();
InputStreamis=newFileInputStream ("test.txt");
int size =is.available ();
for (int i =0; i < size; i++){
System.out.print ((char)is.read ()+"");
}
is.close ();
}catch (IOException e){
System.out.print ("Exception");
}
}
}
```

The above code would create file test.txt and would write given numbers in binary format. Same would be the output on the stdout screen.

File Navigation and I/O

There are several other classes that we would be going through to get to know the basics of File Navigation and I/O.

Java - File Class

Java File class represents the files and directory pathnames in an abstract manner. This class is used for creation of files and directories, file searching, file deletion, etc.

The File object represents the actual file/directory on the disk. Following is the list of constructors to create a File object.

Sr.No.	Method & Description
1	File (File parent, String child) This constructor creates a new File instance from a parent abstract pathname and a child pathname string.
2	File (String pathname) This constructor creates a new File instance by converting the given pathname string into an abstract pathname.
3	File (String parent, String child) This constructor creates a new File instance from a parent pathname string and a child pathname string.
4	File (URI uri) This constructor creates a new File instance by converting the given file: URI into an abstract pathname.

Once you haveFileobject in hand, then there is a list of helper methods which can be used to manipulate the files.

Sr.No.	Method & Description
1	public String getName () Returns the name of the file or directory denoted by this abstract pathname.
2	public String getParent () Returns the pathname string of this abstract pathname's parent, or null if this pathname does not name a parent directory.
3	public File getParentFile () Returns the abstract pathname of this abstract pathname's parent, or null if this pathname does not name a parent directory.
4	public String getPath () Converts this abstract pathname into a pathname string.
5	public boolean isAbsolute () Tests whether this abstract pathname is absolute. Returns true if this abstract pathname is absolute, false otherwise.
6	public String getAbsolutePath () Returns the absolute pathname string of this abstract pathname.
7	public boolean canRead () Tests whether the application can read the file denoted by this abstract pathname. Returns true if and only if the file specified by this abstract pathname exists and can be read by the application; false otherwise.
8	public boolean canWrite () Tests whether the application can modify to the file denoted by this abstract pathname. Returns true if and only if the file system actually contains a file denoted by this abstract pathname and the application is allowed to write to the file; false otherwise.
9	public boolean exists () Tests whether the file or directory denoted by this abstract pathname exists. Returns true if and only if the file or directory denoted by this abstract pathname exists; false otherwise.
10	public boolean isDirectory () Tests whether the file denoted by this abstract pathname is a directory.

	Returns true if and only if the file denoted by this abstract pathname exists and is a directory; false otherwise.
11	public boolean isFile () Tests whether the file denoted by this abstract pathname is a normal file. A file is normal if it is not a directory and, in addition, satisfies other system-dependent criteria. Any non-directory file created by a Java application is guaranteed to be a normal file. Returns true if and only if the file denoted by this abstract pathname exists and is a normal file; false otherwise.
12	public long lastModified () Returns the time that the file denoted by this abstract pathname was last modified. Returns a long value representing the time the file was last modified, measured in milliseconds since the epoch (00:00:00 GMT, January 1, 1970), or 0L if the file does not exist or if an I/O error occurs.
13	public long length () Returns the length of the file denoted by this abstract pathname. The return value is unspecified if this pathname denotes a directory.
14	public boolean createNewFile () throws IOException Atomically creates a new, empty file named by this abstract pathname if and only if a file with this name does not yet exist. Returns true if the named file does not exist and was successfully created; false if the named file already exists.
15	public boolean delete () Deletes the file or directory denoted by this abstract pathname. If this pathname denotes a directory, then the directory must be empty in order to be deleted. Returns true if and only if the file or directory is successfully deleted; false otherwise.
16	public void deleteOnExit () Requests that the file or directory denoted by this abstract pathname be deleted when the virtual machine terminates.
17	public String [] list () Returns an array of strings naming the files and directories in the directory denoted by this abstract pathname.
18	public String [] list (FilenameFilter filter) Returns an array of strings naming the files and directories in the directory denoted by this abstract pathname that satisfy the specified filter.
20	public File [] listFiles () Returns an array of abstract pathnames denoting the files in the directory denoted by this abstract pathname.
21	public File [] listFiles (FileFilter filter) Returns an array of abstract pathnames denoting the files and directories in the directory denoted by this abstract pathname that satisfy the specified filter.
22	public boolean mkdir () Creates the directory named by this abstract pathname. Returns true if and only if the directory was created; false otherwise.
23	public boolean mkdirs () Creates the directory named by this abstract pathname, including any necessary but nonexistent parent directories. Returns true if and only if the directory was created, along with all necessary parent directories; false otherwise.
24	public boolean renameTo (File dest) Renames the file denoted by this abstract pathname. Returns true if and only if the renaming succeeded; false otherwise.
25	public boolean setLastModified (long time) Sets the last-modified time of the file or directory named by this abstract pathname. Returns true if and only if the operation succeeded; false otherwise.
26	public boolean setReadOnly () Marks the file or directory named by this abstract pathname so that only read operations are allowed. Returns true if and only if the operation succeeded; false otherwise.
27	public static File createTempFile (String prefix, String suffix, File directory) throws IOException Creates a new empty file in the specified directory, using the given prefix and suffix strings to generate its name. Returns an abstract pathname denoting a newly-created empty file.
28	public static File createTempFile (String prefix, String suffix) throws IOException Creates an empty file in the default temporary-file directory, using the given prefix and suffix to generate its name. Invoking this method is equivalent to invoking createTempFile (prefix, suffix, null). Returns abstract pathname denoting a newly-created empty file.
29	public int compareTo (File pathname) Compares two abstract pathnames lexicographically. Returns zero if the argument is equal to this abstract pathname, a value less than zero if this abstract pathname is lexicographically less than the argument, or a value greater than zero if this abstract pathname is lexicographically greater than the argument.
30	public int compareTo (Object o) Compares this abstract pathname to another object. Returns zero if the argument is equal to this abstract pathname, a value less than zero if this abstract pathname is lexicographically less than the argument, or a value greater than zero if this abstract pathname is lexicographically greater than the argument.
31	public boolean equals (Object obj) Tests this abstract pathname for equality with the given object. Returns true if and only if the argument is not null and is an abstract pathname that denotes the same file or directory as this abstract pathname.
32	public String toString () Returns the pathname string of this abstract pathname. This is just the string returned by the getPath () method.

Example

Following is an example to demonstrate File object

```
import java.io.File;
publicclassFileDemo{
publicstaticvoid main (String [] args){
File f =null;
String [] strs ={"test1.txt","test2.txt"};
try{
// for each string in string array
for (String s:strs ){
// create new file
f =newFile (s);
// true if the file is executable
booleanbool= f.canExecute ();
// find the absolute path
String a = f.getAbsolutePath ();
// prints absolute path
System.out.print (a);
// prints
System.out.println (" is executable: "+bool);
}
}catch (Exception e){
// if any I/O error occurs
e.printStackTrace ();
}
}
}
```

Consider there is an executable file test1.txt and another file test2.txt is non executable in the current directory. Let us compile and run the above program, This will produce the following result

Output

```
/home/cg/root/2880380/test1.txt is executable: false
/home/cg/root/2880380/test2.txt is executable: false
```

Java - FileReader Class

This class inherits from the InputStreamReader class. FileReader is used for reading streams of characters.

This class has several constructors to create required objects. Following is the list of constructors provided by the FileReader class.

Sr.No.	Constructor & Description
1	FileReader (File file) This constructor creates a new FileReader, given the File to read from.
2	FileReader (FileDescriptor fd) This constructor creates a new FileReader, given the FileDescriptor to read from.
3	FileReader (String fileName) This constructor creates a new FileReader, given the name of the file to read from.

Once you have FileReader object in hand then there is a list of helper methods which can be used to manipulate the files.

Sr.No.	Method & Description
1	public int read () throws IOException Reads a single character. Returns an int, which represents the character read.
2	public int read (char [] c, int offset, int len) Reads characters into an array. Returns the number of characters read.

Example

Following is an example to demonstrate class

```
import java.io.*;
publicclassFileRead{
publicstaticvoid main (String args [])throwsIOException{
File file =newFile ("Hello1.txt");
// creates the file
file.createNewFile ();
// creates a FileWriter Object
FileWriter writer =newFileWriter (file);
// Writes the content to the file
writer.write ("This\n is\n an\n example\n");
writer.flush ();
writer.close ();
// Creates a FileReader Object
FileReader fr =newFileReader (file);
char [] a =newchar [50];
fr.read (a);// reads the content to the array
for (char c: a)
System.out.print (c);// prints the characters one by one
fr.close ();
}
}
```

This will produce the following result

Output

```
This
is
an
example
```

Java - FileWriter Class

This class inherits from the OutputStreamWriter class. The class is used for writing streams of characters.

This class has several constructors to create required objects. Following is a list.

Once you haveFileWriterobject in hand, then there is a list of helper methods, which can be used to manipulate the files.

Sr.No.	Constructor & Description
1	FileWriter (File file) This constructor creates a FileWriter object given a File object.
2	FileWriter (File file, boolean append) This constructor creates a FileWriter object given a File object with a boolean indicating whether or not to append the data written.
3	FileWriter (FileDescriptor fd) This constructor creates a FileWriter object associated with the given file descriptor.
4	FileWriter (String fileName) This constructor creates a FileWriter object, given a file name.
5	FileWriter (String fileName, boolean append) This constructor creates a FileWriter object given a file name with a boolean indicating whether or not to append the data written.

Sr.No.	Method & Description
1	public void write (int c) throws IOException Writes a single character.
2	public void write (char [] c, int offset, int len) Writes a portion of an array of characters starting from offset and with a length of len.
3	public void write (String s, int offset, int len) Write a portion of a String starting from offset and with a length of len.

Example

Following is an example to demonstrate class

```
import java.io.*;
publicclassFileRead{
publicstaticvoid main (String args []) throwsIOException{
File file =newFile ("Hello1.txt");
// creates the file
file.createNewFile ();
// creates a FileWriter Object
FileWriter writer =newFileWriter (file);
// Writes the content to the file
writer.write ("This\n is\n an\n example\n");
writer.flush ();
writer.close ();
// Creates a FileReader Object
FileReader fr =newFileReader (file);
char [] a =newchar [50];
fr.read (a);// reads the content to the array
for (char c: a)
System.out.print (c);// prints the characters one by one
fr.close ();
}
}
```

This will produce the following result

Output

```
This
is
an
example
```

Java Exceptions

An exception (or exceptional event) is a problem that arises during the execution of a program. When anExceptionoccurs the normal flow of the program is disrupted and the program/Application terminates abnormally, which is not recommended, therefore, these exceptions are to be handled.

An exception can occur for many different reasons. Following are some scenarios where an exception occurs.

A user has entered an invalid data.

A file that needs to be opened cannot be found.

A network connection has been lost in the middle of communications or the JVM has run out of memory.

Some of these exceptions are caused by user error, others by programmer error, and others by physical resources that have failed in some manner.

Based on these, we have three categories of Exceptions. You need to understand them to know how exception handling works in Java.

Checked exceptions A checked exception is an exception that occurs at the compile time, these are also called as compile time exceptions. These exceptions cannot simply be ignored at the time of compilation, the programmer should take care of (handle) these exceptions.

For example, if you use FileReader class in your program to read data from a file, if the file specified in its constructor doesn't exist, then a FileNotFoundException occurs, and the compiler prompts the programmer to handle the exception.

Example

```
import java.io.File;
import java.io.FileReader;
publicclassFilenotFound_Demo{
publicstaticvoid main (String args []){
File file =newFile ("E://file.txt");
FileReader fr =newFileReader (file);
}
}
```

If you try to compile the above program, you will get the following exceptions.

Output

```
C:\>javac FilenotFound_Demo.java
FilenotFound_Demo.java:8: error: unreported exception FileNotFoundException; must be caught or declared to be thrown
FileReader fr = new FileReader (file);
^
1 error
```

Note Since the methods read () and close () of FileReader class throws IOException, you can observe that the compiler notifies to handle IOException, along with FileNotFoundException.

Unchecked exceptions An unchecked exception is an exception that occurs at the time of execution. These are also called as Runtime Exceptions. These include programming bugs, such as logic errors or improper use of an API. Runtime exceptions are ignored at the time of compilation.

For example, if you have declared an array of size 5 in your program, and trying to call the 6th element of the array then an ArrayIndexOutOfBoundsExceptionexception occurs.

Example

```
publicclassUnchecked_Demo{
publicstaticvoid main (String args []){
int num []={1,2,3,4};
System.out.println (num [5]);
}
}
```

If you compile and execute the above program, you will get the following exception.

Output

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
at Exceptions.Unchecked_Demo.main (Unchecked_Demo.java:8)
```

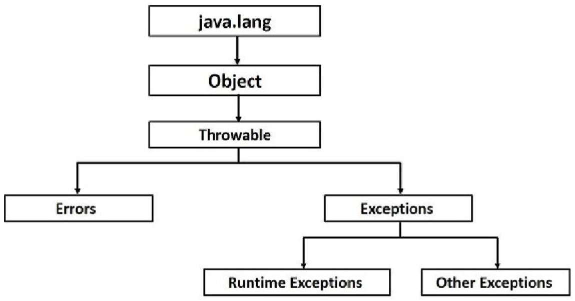
Errors These are not exceptions at all, but problems that arise beyond the control of the user or the programmer. Errors are typically ignored in your code because you can rarely do anything about an error. For example, if a stack overflow occurs, an error will arise. They are also ignored at the time of compilation.

Exception Hierarchy

All exception classes are subtypes of the java.lang.Exception class. The exception class is a subclass of the Throwable class. Other than the exception class there is another subclass called Error which is derived from the Throwable class.

Errors are abnormal conditions that happen in case of severe failures, these are not handled by the Java programs. Errors are generated to indicate errors generated by the runtime environment. Example: JVM is out of memory. Normally, programs cannot recover from errors.

The Exception class has two main subclasses: IOException class and RuntimeException Class.



Following is a list of most common checked and uncheckedJava's Built-in Exceptions.

Exceptions Methods

Following is the list of important methods available in the Throwable class.

Sr.No.	Method & Description
1	public String getMessage () Returns a detailed message about the exception that has occurred. This message is initialized in the Throwable constructor.
2	public Throwable getCause () Returns the cause of the exception as represented by a Throwable object.
3	public String toString () Returns the name of the class concatenated with the result of getMessage ().
4	public void printStackTrace () Prints the result of toString () along with the stack trace to System.err, the error output stream.
5	public StackTraceElement [] getStackTrace () Returns an array containing each element on the stack trace. The element at index 0 represents the top of the call stack, and the last element in the array represents the method at the bottom of the call stack.
6	public Throwable fillInStackTrace () Fills the stack trace of this Throwable object with the current stack trace, adding to any previous information in the stack trace.

Catching Exceptions

A method catches an exception using a combination of the try and catch keywords. A try/catch block is placed around the code that might generate an exception. Code within a try/catch block is referred to as protected code, and the syntax for using try/catch looks like the following

Syntax

```
try {
// Protected code
} catch (ExceptionName e1) {
// Catch block
}
```

The code which is prone to exceptions is placed in the try block. When an exception occurs, that exception occurred is handled by catch block associated with it. Every try block should be immediately followed either by a catch block or finally block.

A catch statement involves declaring the type of exception you are trying to catch. If an exception occurs in protected code, the catch block (or blocks) that follows the try is checked. If the type of exception that occurred is listed in a catch block, the exception is passed to the catch block much as an argument is passed into a method parameter.

Example

The following is an array declared with 2 elements. Then the code tries to access the 3rd element of the array which throws an exception.

```
// File Name: ExcepTest.java
import java.io.*;
publicclassExcepTest{
publicstaticvoid main (String args []){
try{
int a []=newint [2];
System.out.println ("Access element three:"+ a
[3]);
}catch (ArrayIndexOutOfBoundsException e){
System.out.println ("Exception thrown:"+ e);
}
System.out.println ("Out of the block");
}
}
```

This will produce the following result

Output

```
Exception
thrown:java.lang.ArrayIndexOutOfBoundsException:
3
Out of the block
```

Multiple Catch Blocks

A try block can be followed by multiple catch blocks. The syntax for multiple catch blocks looks like the following

Syntax

```
try {
// Protected code
} catch (ExceptionType1 e1) {
// Catch block
} catch (ExceptionType2 e2) {
// Catch block
} catch (ExceptionType3 e3) {
// Catch block
}
```

The previous statements demonstrate three catch blocks, but you can have any number of them after a single try. If an exception occurs in the protected code, the exception is thrown to the first catch block in the list. If the data type of the exception thrown matches ExceptionType1, it gets caught there. If not, the exception passes down to the second catch statement. This continues until the exception either is caught or falls through all catches, in which case the current method stops execution and the exception is thrown down to the previous method on the call stack.

Example

Here is code segment showing how to use multiple try/catch statements.

```
try{
file =newFileInputStream (fileName);
x =(byte) file.read ();
}catch (IOException i){
i.printStackTrace ();
return-1;
}catch (FileNotFoundException f)// Not valid! {
f.printStackTrace ();
return-1;
}
```

Catching Multiple Type of Exceptions

Since Java 7, you can handle more than one exception using a single catch block, this feature simplifies the code. Here is how you would do it

```
catch (IOException|FileNotFoundException ex)
{
logger.log (ex);
throw ex;
```

The Throws/Throw Keywords

If a method does not handle a checked exception, the method must declare it using thethrowskeyword. The throws keyword appears at the end of a method's signature.

You can throw an exception, either a newly instantiated one or an exception that you just caught, by using thethrowkeyword.

Try to understand the difference between throws and throw keywords,throwsis used to postpone the handling of a checked exception andthrowis used to invoke an exception explicitly.

The following method declares that it throws a RemoteException

Example

```
import java.io.*;
publicclass className {
publicvoid deposit (double amount)throwsRemoteException{
// Method implementation
thrownewRemoteException ();
}
// Remainder of class definition
}
```

A method can declare that it throws more than one exception, in which case the exceptions are declared in a list separated by commas. For example, the following method declares that it throws a RemoteException and an InsufficientFundsException

Example

```
import java.io.*;
publicclass className {
publicvoid withdraw (double amount)throwsRemoteException,
InsufficientFundsException{
// Method implementation
}
// Remainder of class definition
}
```

The Finally Block

The finally block follows a try block or a catch block. A finally block of code always executes, irrespective of occurrence of an Exception.

Using a finally block allows you to run any cleanup-type statements that you want to execute, no matter what happens in the protected code.

A finally block appears at the end of the catch blocks and has the following syntax

Syntax

```
try {
// Protected code
} catch (ExceptionType1 e1) {
// Catch block
} catch (ExceptionType2 e2) {
// Catch block
} catch (ExceptionType3 e3) {
// Catch block
}finally {
// The finally block always executes.
}
```

Example

```
publicclassExcepTest{
publicstaticvoid main (String args []){
int a []=newint [2];
try{
System.out.println ("Access element three:"+ a [3]);
}catch (ArrayIndexOutOfBoundsException e){
System.out.println ("Exception thrown:"+ e);
}finally{
a [0]=6;
System.out.println ("First element value: "+ a [0]);
System.out.println ("The finally statement is
executed");
}
}
}
```

This will produce the following result

Output

```
Exception
thrown:java.lang.ArrayIndexOutOfBoundsException:
3
First element value: 6
The finally statement is executed
```

Note the following

A catch clause cannot exist without a try statement.

It is not compulsory to have finally clauses whenever a try/catch block is present.

The try block cannot be present without either catch clause or finally clause.

Any code cannot be present in between the try, catch, finally blocks.

The try-with-resources

Generally, when we use any resources like streams, connections, etc. we have to close them explicitly using finally block. In the following program, we are reading data from a file usingFileReaderand we are closing it using finally block.

Example

```
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
publicclassReadData_Demo{
publicstaticvoid main (String args []){
FileReader fr =null;
try{
File file =newFile ("file.txt");
fr =newFileReader (file);char [] a =newchar
[50];
fr.read (a);// reads the content to the array
for (char c: a)
System.out.print (c);// prints the characters one
by one
}catch (IOException e){
e.printStackTrace ();
}finally{
try{
fr.close ();
}catch (IOException ex){
ex.printStackTrace ();
}
}
}
}
```

try-with-resources, also referred asautomatic resource management, is a new exception handling mechanism that was introduced in Java 7, which automatically closes the resources used within the try catch block.

To use this statement, you simply need to declare the required resources within the parenthesis, and the created resource will be closed automatically at the end of the block. Following is the syntax of try-with-resources statement.

Syntax

```
try (FileReader fr = new FileReader ("file path"))
{
// use the resource
} catch () {
// body of catch
}
}
```

Following is the program that reads the data in a file using try-with-resources statement.

Example

```
import java.io.FileReader;
import java.io.IOException;
publicclassTry_withDemo{
publicstaticvoid main (String args []){
try (FileReader fr =newFileReader
("E://file.txt")){
char [] a =newchar [50];
fr.read (a);// reads the contentto the array
for (char c: a)
System.out.print (c);// prints the characters one
by one
}catch (IOException e){
e.printStackTrace ();
}
}
}
```

Following points are to be kept in mind while working with try-with-resources statement.

To use a class with try-with-resources statement it should implementAutoCloseableinterface and theclose ()method of it gets invoked automatically at runtime.

You can declare more than one class in try-with-resources statement.

While you declare multiple classes in the try block of try-with-resources statement these classes are closed in reverse order.

Except the declaration of resources within the parenthesis everything is the same as normal try/catch block of a try block.

The resource declared in try gets instantiated just before the start of the try-block.

The resource declared at the try block is implicitly declared as final.

User-defined Exceptions

You can create your own exceptions in Java. Keep the following points in mind when writing your own exception classes

All exceptions must be a child of Throwable.

If you want to write a checked exception that is automatically enforced by the Handle or Declare Rule, you need to extend the Exception class.

If you want to write a runtime exception, you need to extend the RuntimeException class.

We can define our own Exception class as below

```
class MyException extends Exception {
}
```

You just need to extend the predefinedExceptionclass to create your own Exception. These are considered to be checked exceptions. The followingInsufficientFundsExceptionclass is a user-defined exception that extends the Exception class, making it a checked exception. An exception class is like any other class, containing useful fields and methods.

Example

```
// File Name InsufficientFundsException.java
import java.io.*;
publicclassInsufficientFundsExceptionextendsException{
privatedouble amount;
publicInsufficientFundsException (double amount){
this.amount = amount;
}
publicdouble getAmount (){
return amount;
}
}
```

To demonstrate using our user-defined exception, the following CheckingAccount class contains a withdraw () method that throws an InsufficientFundsException.

```
// File Name CheckingAccount.java
import java.io.*;
publicclassCheckingAccount{
privatedouble balance;
privateint number;
publicCheckingAccount (int number){
this.number = number;
}
publicvoid deposit (double amount){
balance += amount;
}
publicvoid withdraw (double amount)throwsInsufficientFundsException{
if (amount <= balance){
balance -= amount;
}else{
double needs = amount - balance;
thrownewInsufficientFundsException (needs);
}
}
publicdouble getBalance (){
return balance;
}
publicint getNumber (){
return number;
}
}
```

The following BankDemo program demonstrates invoking the deposit () and withdraw () methods of CheckingAccount.

```
// File Name BankDemo.java
publicclassBankDemo{
publicstaticvoid main (String [] args){
CheckingAccount c =newCheckingAccount (101);
System.out.println ("Depositing $500...");
c.deposit (500.00);
try{
System.out.println ("\nWithdrawing $100...");
c.withdraw (100.00);
System.out.println ("\nWithdrawing $600...");
c.withdraw (600.00);
}catch (InsufficientFundsException e){
System.out.println ("Sorry, but you are short $"+ e.getAmount ());
e.printStackTrace ();
}
}
}
```

Compile all the above three files and run BankDemo. This will produce the following result

Output

```
Depositing $500...
Withdrawing $100...
Withdrawing $600...
Sorry, but you are short $200.0
InsufficientFundsException
at CheckingAccount.withdraw (CheckingAccount.java:25)
at BankDemo.main (BankDemo.java:13)
```

Common Exceptions

In Java, it is possible to define two catergories of Exceptions and Errors.

JVM Exceptions These are exceptions/errors that are exclusively or logically thrown by the JVM. Examples: NullPointerException, ArrayIndexOutOfBoundsException, ClassCastException.

Programmatic Exceptions These exceptions are thrown explicitly by the application or the API programmers. Examples: IllegalArgumentException, IllegalStateException.

Java - Inner classes

In this chapter, we will discuss inner classes of Java.

Nested Classes

In Java, just like methods, variables of a class too can have another class as its member. Writing a class within another is allowed in Java. The class written within is called thenested class, and the class that holds the inner class is called theouter class.

Syntax

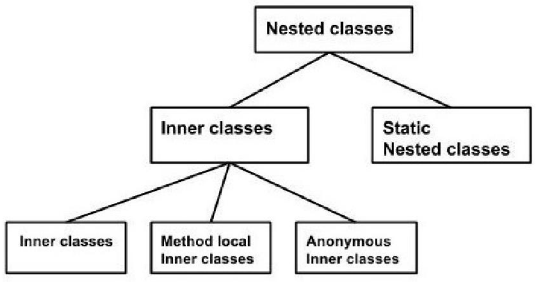
Following is the syntax to write a nested class. Here, the classOuter_Demois the outer class and the classInner_Demois the nested class.

```
class Outer_Demo {
class Nested_Demo {
}
}
```

Nested classes are divided into two types

Non-static nested classes These are the non-static members of a class.

Static nested classes These are the static members of a class.



Inner Classes (Non-static Nested Classes)

Inner classes are a security mechanism in Java. We know a class cannot be associated with the access modifierprivate, but if we have the class as a member of other class, then the inner class can be made private. And this is also used to access the private members of a class.

Inner classes are of three types depending on how and where you define them. They are

Inner Class

Method-local Inner Class

Anonymous Inner Class

Inner Class

Creating an inner class is quite simple. You just need to write a class within a class. Unlike a class, an inner class can be private and once you declare an inner class private, it cannot be accessed from an object outside the class.

Following is the program to create an inner class and access it. In the given example, we make the inner class private and access the class through a method.

Example

```
classOuter_Demo{
int num;
// inner class
privateclassInner_Demo{
publicvoidprint (){
System.out.println ("This is an inner class");
}
}
// Accessing he inner class from the method within
void display_Inner (){
Inner_Demo inner =newInner_Demo ();
inner.print ();
}
}
publicclassMy_class{
publicstaticvoid main (String args []){
// Instantiating the outer class
Outer_Demo outer =newOuter_Demo ();
// Accessing the display_Inner () method.
outer.display_Inner ();
}
}
```

Here you can observe thatOuter_Demois the outer class,Inner_Demois the inner class,display_Inner ()is the method inside which we are instantiating the inner class, and this method is invoked from themainmethod.

If you compile and execute the above program, you will get the following result

Output

```
This is an inner class.
```

Accessing the Private Members

As mentioned earlier, inner classes are also used to access the private members of a class. Suppose, a class is having private members to access them. Write an inner class in it, return the private members from a method within the inner class, say,getValue (), and finally from another class (from which you want to access the private members) call the getValue () method of the inner class.

To instantiate the inner class, initially you have to instantiate the outer class. Thereafter, using the object of the outer class, following is the way in which you can instantiate the inner class.

```
Outer_Demo outer = new Outer_Demo ();
Outer_Demo.Inner_Demo inner = outer.new Inner_Demo ();
```

The following program shows how to access the private members of a class using inner class.

Example

```
classOuter_Demo{
// private variable of the outer class
privateint num =175;
// inner class
publicclassInner_Demo{
publicint getNum (){
System.out.println ("This is the getnum method
of the inner class");
return num;
}
}
}
publicclassMy_class2{
publicstaticvoid main (String args []){
// Instantiating the outer class
Outer_Demo outer =newOuter_Demo ();
// Instantiating the inner class
Outer_Demo.Inner_Demo inner = outer.newInner_Demo ();
System.out.println (inner.getNum ());
}
}
```

If you compile and execute the above program, you will get the following result

Output

```
This is the getnum method of the inner class: 175
```

Method-local Inner Class

In Java, we can write a class within a method and this will be a local type. Like local variables, the scope of the inner class is restricted within the method.

A method-local inner class can be instantiated only within the method where the inner class is defined. The following program shows how to use a method-local inner class.

Example

```
publicclassOuterclass{
// instance method of the outer class
void my_Method (){
int num =23;
// method-local inner class
classMethodInner_Demo{
publicvoidprint (){
System.out.println ("This is method inner class "+num);
}
}// end of inner class
// Accessing the inner class
MethodInner_Demo inner =newMethodInner_Demo ();
inner.print ();
}
publicstaticvoid main (String args []){
Outerclass outer =newOuterclass ();
outer.my_Method ();
}
}
```

If you compile and execute the above program, you will get the following result

Output

```
This is method inner class 23
```

Anonymous Inner Class

An inner class declared without a class name is known as ananonymous inner class. In case of anonymous inner classes, we declare and instantiate them at the same time. Generally, they are used whenever you need to override the method of a class or an interface. The syntax of an anonymous inner class is as follows

Syntax

```
AnonymousInner an_inner = new AnonymousInner () {
public void my_method () {
........
........
}
};
```

The following program shows how to override the method of a class using anonymous inner class.

Example

```
abstractclassAnonymousInner{
publicabstractvoid mymethod ();
}
publicclassOuter_class{
publicstaticvoid main (String args []){
AnonymousInner inner =newAnonymousInner (){
publicvoid mymethod (){
System.out.println ("This is an example of anonymous inner class");
}
};
inner.mymethod ();
}
}
```

If you compile and execute the above program, you will get the following result

Output

```
This is an example of anonymous inner class
```

In the same way, you can override the methods of the concrete class as well as the interface using an anonymous inner class.

Anonymous Inner Class as Argument

Generally, if a method accepts an object of an interface, an abstract class, or a concrete class, then we can implement the interface, extend the abstract class, and pass the object to the method. If it is a class, then we can directly pass it to the method.

But in all the three cases, you can pass an anonymous inner class to the method. Here is the syntax of passing an anonymous inner class as a method argument

```
obj.my_Method (new My_Class () {
public void Do () {
.....
.....
}
});
```

The following program shows how to pass an anonymous inner class as a method argument.

Example

```
// interface
interfaceMessage{
String greet ();
}
publicclassMy_class{
// method which accepts the object of interface Message
publicvoid displayMessage (Message m){
System.out.println (m.greet ()+
", This is an example of anonymous inner class as an argument");
}
publicstaticvoid main (String args []){
// Instantiating the class
My_class obj =newMy_class ();
// Passing an anonymous inner class as an argument
obj.displayMessage (newMessage (){
publicString greet (){
return"Hello";
}
});
}
}
```

If you compile and execute the above program, it gives you the following result

Output

```
Hello, This is an example of anonymous inner class as an argument
```

Static Nested Class

A static inner class is a nested class which is a static member of the outer class. It can be accessed without instantiating the outer class, using other static members. Just like static members, a static nested class does not have access to the instance variables and methods of the outer class. The syntax of static nested class is as follows

Syntax

```
class MyOuter {
static class Nested_Demo {
}
}
```

Instantiating a static nested class is a bit different from instantiating an inner class. The following program shows how to use a static nested class.

Example

```
publicclassOuter{
staticclassNested_Demo{
publicvoid my_method (){
System.out.println ("This is my nested class");
}
}
publicstaticvoid main (String args []){
Outer.Nested_Demo nested =newOuter.Nested_Demo ();
nested.my_method ();
}
}
```

If you compile and execute the above program, you will get the following result

Output

```
This is my nested class
```

Java Inheritance

Inheritance can be defined as the process where one class acquires the properties (methods and fields) of another. With the use of inheritance the information is made manageable in a hierarchical order.

The class which inherits the properties of other is known as subclass (derived class, child class) and the class whose properties are inherited is known as superclass (base class, parent class).

extends Keyword

extendsis the keyword used to inherit the properties of a class. Following is the syntax of extends keyword.

Syntax

```
class Super {
.....
.....
}
class Sub extends Super {
.....
.....
}
```

Sample Code

Following is an example demonstrating Java inheritance. In this example, you can observe two classes namely Calculation and My_Calculation.

Using extends keyword, the My_Calculation inherits the methods addition () and Subtraction () of Calculation class.

Copy and paste the following program in a file with name My_Calculation.java

Example

```
classCalculation{
int z;
publicvoid addition (int x,int y){
z = x + y;
System.out.println ("The sum of the given numbers:"+z);
}
publicvoidSubtraction (int x,int y){
z = x - y;
System.out.println ("The difference between the given numbers:"+z);
}
}
```

publicclassMy_CalculationextendsCalculation{
publicvoid multiplication (int x,int y){
z = x * y;
System.out.println ("The product of the given numbers:"+z);
}
publicstaticvoid main (String args []){
int a =20, b =10;
My_Calculation demo =newMy_Calculation ();
demo.addition (a, b);
demo.Subtraction (a, b);
demo.multiplication (a, b);
}
}

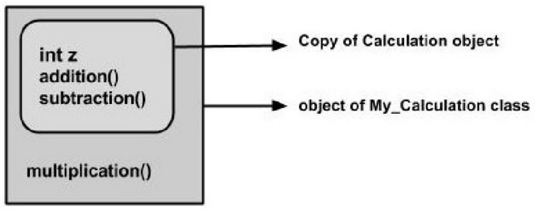
Compile and execute the above code as shown below.

javac My_Calculation.java
java My_Calculation

After executing the program, it will produce the following result

Output
The sum of the given numbers:30
The difference between the given numbers:10
The product of the given numbers:200

In the given program, when an object toMy_Calculationclass is created, a copy of the contents of the superclass is made within it. That is why, using the object of the subclass you can access the members of a superclass.



The Superclass reference variable can hold the subclass object, but using that variable you can access only the members of the superclass, so to access the members of both classes it is recommended to always create reference variable to the subclass.

If you consider the above program, you can instantiate the class as given below. But using the superclass reference variable (calin this case) you cannot call the methodmultiplication (), which belongs to the subclass My_Calculation.

Calculation cal =newMy_Calculation ();
demo.addition (a, b);
demo.Subtraction (a, b);

Note A subclass inherits all the members (fields, methods, and nested classes) from its superclass. Constructors are not members, so they are not inherited by subclasses, but the constructor of the superclass can be invoked from the subclass.

The super keyword

Thesuperkeyword is similar tothiskeyword. Following are the scenarios where the super keyword is used.

It is used todifferentiate the membersof superclass from the members of subclass, if they have same names.

It is used toinvoke the superclassconstructor from subclass.

Differentiating the Members

If a class is inheriting the properties of another class. And if the members of the superclass have the names same as the sub class, to differentiate these variables we use super keyword as shown below.

super.variable
super.method ();

Sample Code

This section provides you a program that demonstrates the usage of thesuperkeyword.

In the given program, you have two classes namelySub_classandSuper_class, both have a method named display () with different implementations, and a variable named num with different values. We are invoking display () method of both classes and printing the value of the variable num of both classes. Here you can observe that we have used super keyword to differentiate the members of superclass from subclass.

Copy and paste the program in a file with name Sub_class.java.

Example

classSuper_class{
int num =20;
// display method of superclass
publicvoid display (){
System.out.println ("This is the display method of superclass");
}
}
publicclassSub_classextendsSuper_class{
int num =10;
// display method of sub class
publicvoid display (){
System.out.println ("This is the display method of subclass");
}
publicvoid my_method (){
// Instantiating subclass
Sub_classsub=newSub_class ();
// Invoking the display () method of sub class
sub.display ();
// Invoking the display () method of superclass
super.display ();
// printing the value of variable num of subclass
System.out.println ("value of the variable

```
named num in sub class:"+sub.num);
// printing the value of variable num of superclass
System.out.println ("value of the variable named num in super class:"+super.num);
}
publicstaticvoid main (String args []){
Sub_class obj =newSub_class ();
obj.my_method ();
}
}
```

Compile and execute the above code using the following syntax.

```
javac Super_Demo
java Super
```

On executing the program, you will get the following result

Output

```
This is the display method of subclass
This is the display method of superclass
value of the variable named num in sub class:10
value of the variable named num in super class:20
```

Invoking Superclass Constructor

If a class is inheriting the properties of another class, the subclass automatically acquires the default constructor of the superclass. But if you want to call a parameterized constructor of the superclass, you need to use the super keyword as shown below.

```
super (values);
```

Sample Code

The program given in this section demonstrates how to use the super keyword to invoke the parametrized constructor of the superclass. This program contains a superclass and a subclass, where the superclass contains a parameterized constructor which accepts a string value, and we used the super keyword to invoke the parameterized constructor of the superclass.

Copy and paste the following program in a file with the name Subclass.java

Example

```
classSuperclass{
int age;
Superclass (int age){
this.age = age;
}
publicvoid getAge (){
System.out.println ("The value of the variable named age in super class is: "+age);
}
}
publicclassSubclassextendsSuperclass{
Subclass (int age){
super (age);
}
publicstaticvoid main (String argd []){
Subclass s =newSubclass (24);
s.getAge ();
}
}
```

Compile and execute the above code using the following syntax.

```
javac Subclass
java Subclass
```

On executing the program, you will get the following result

Output

```
The value of the variable named age in super class is: 24
```

IS-A Relationship

IS-A is a way of saying: This object is a type of that object. Let us see how theextendskeyword is used to achieve inheritance.

```
publicclassAnimal{
}
publicclassMammalextendsAnimal{
}
publicclassReptileextendsAnimal{
}
publicclassDogextendsMammal{
}
```

Now, based on the above example, in Object-Oriented terms, the following are true

Animal is the superclass of Mammal class.

Animal is the superclass of Reptile class.

Mammal and Reptile are subclasses of Animal class.

Dog is the subclass of both Mammal and Animal classes.

Now, if we consider the IS-A relationship, we can say

Mammal IS-A Animal

Reptile IS-A Animal

Dog IS-A Mammal

Hence: Dog IS-A Animal as well

With the use of the extends keyword, the subclasses will be able to inherit all the properties of the superclass except for the private properties of the superclass.

We can assure that Mammal is actually an Animal with the use of the instance operator.

Example

```
classAnimal{
}
classMammalextendsAnimal{
}
classReptileextendsAnimal{
}
publicclassDogextendsMammal{
publicstaticvoid main (String args []){
Animal a =newAnimal ();
Mammal m =newMammal ();
Dog d =newDog ();
System.out.println (m instanceofAnimal);
System.out.println (d instanceofMammal);
System.out.println (d instanceofAnimal);
}
}
```

This will produce the following result

Output

```
true
true
true
```

Since we have a good understanding of theextendskeyword, let us look into how theimplementskeyword is used to get the IS-A relationship.

Generally, theimplementskeyword is used with classes to inherit the properties of an interface. Interfaces can never be extended by a class.

Example

```
publicinterfaceAnimal{
}
publicclassMammalimplementsAnimal{
}
publicclassDogextendsMammal{
}
```

The instanceof Keyword

Let us use theinstanceofoperator to check determine whether Mammal is actually an Animal, and dog is actually an Animal.

Example

```
interfaceAnimal{}
classMammalimplementsAnimal{}
publicclassDogextendsMammal{
publicstaticvoid main (String args []){
Mammal m =newMammal ();
Dog d =newDog ();
System.out.println (m instanceofAnimal);
System.out.println (d instanceofMammal);
System.out.println (d instanceofAnimal);
}
}
```

This will produce the following result

Output

```
true
true
true
```

HAS-A relationship

These relationships are mainly based on the usage. This determines whether a certain classHAS-Acertain thing. This relationship helps to reduce duplication of code as well as bugs.

Lets look into an example

Example

```
publicclassVehicle{}
publicclassSpeed{}
publicclassVanextendsVehicle{
privateSpeed sp;
}
```

This shows that class Van HAS-A Speed. By having a separate class for Speed, we do not have to put the entire code that belongs to speed inside the Van class, which makes it possible to reuse the Speed class in multiple applications.

In Object-Oriented feature, the users do not need to bother about which object is doing the real work. To achieve this, the Van class hides the implementation details from the users of the Van class. So, basically what happens is the users would ask the Van class to do a certain action and the Van class will either do the work by itself or ask another class to perform the action.

Types of Inheritance

There are various types of inheritance as demonstrated below.

Java Overriding

In the previous chapter, we talked about superclasses and subclasses. If a class inherits a method from its superclass, then there is a chance to override the method provided that it is not marked final.

The benefit of overriding is: ability to define a behavior that's specific to the subclass type, which means a subclass can implement a parent class method based on its requirement.

In object-oriented terms, overriding means to override the functionality of an existing method.

Example

Let us look at an example.

```
classAnimal{
publicvoid move (){
System.out.println ("Animals can move");
}
}
classDogextendsAnimal{
publicvoid move (){
System.out.println ("Dogs can walk and run");
}
}
publicclassTestDog{
publicstaticvoid main (String args []){
Animal a =newAnimal ();// Animal reference and object
Animal b =newDog ();// Animal reference but Dog object
a.move ();// runs the method in Animal class
b.move ();// runs the method in Dog class
}
}
```

This will produce the following result

Output

```
Animals can move
```

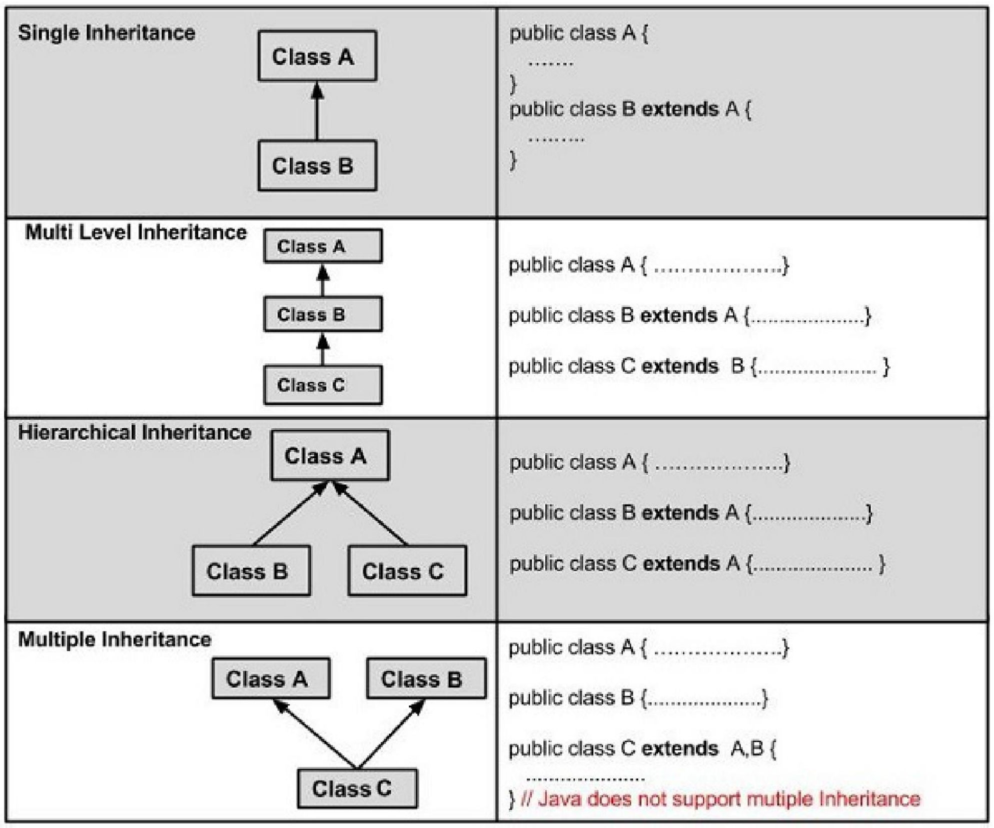
Dogs can walk and run

In the above example, you can see that even thoughbis a type of Animal it runs the move method in the Dog class. The reason for this is: In compile time, the check is made on the reference type. However, in the runtime, JVM figures out the object type and would run the method that belongs to that particular object.

Therefore, in the above example, the program will compile properly since Animal class has the method move. Then, at the runtime, it runs the method specific for that object.

Consider the following example

Example



```
public class A {
  .......
}
public class B extends A {
  .........
}
```

```
public class A { ..................}
public class B extends A {...................}
public class C extends  B {.................... }
```

```
public class A { ..................}
public class B extends A {...................}
public class C extends A {.................... }
```

```
public class A { ..................}
public class B {...................}
public class C extends  A,B {
  ....................
} // Java does not support mutiple Inheritance
```

A very important fact to remember is that Java does not support multiple inheritance. This means that a class cannot extend more than one class. Therefore following is illegal

Example

publicclassextendsAnimal,Mammal{}

However, a class can implement one or more interfaces, which has helped Java get rid of the impossibility of multiple inheritance.

```
classAnimal{
publicvoid move (){
System.out.println ("Animals can move");
}
}
classDogextendsAnimal{
publicvoid move (){
System.out.println ("Dogs can walk and run");
}
publicvoid bark (){
System.out.println ("Dogs can bark");
}
}
publicclassTestDog{
publicstaticvoid main (String args []){
Animal a =newAnimal ();// Animal reference and object
Animal b =newDog ();// Animal reference but Dog object
a.move ();// runs the method in Animal class
b.move ();// runs the method in Dog class
b.bark ();
}
}
```

This will produce the following result

Output

```
TestDog.java:26: error: cannot find symbol
b.bark ();
^
symbol:   method bark ()
location: variable b of type Animal
```

```
1 error
```

This program will throw a compile time error since b's reference type Animal doesn't have a method by the name of bark.

Rules for Method Overriding

The argument list should be exactly the same as that of the overridden method.

The return type should be the same or a subtype of the return type declared in the original overridden method in the superclass.

The access level cannot be more restrictive than the overridden method's access level. For example: If the superclass method is declared public then the overridding method in the sub class cannot be either private or protected.

Instance methods can be overridden only if they are inherited by the subclass.

A method declared final cannot be overridden.

A method declared static cannot be overridden but can be re-declared.

If a method cannot be inherited, then it cannot be overridden.

A subclass within the same package as the instance's superclass can override any superclass method that is not declared private or final.

A subclass in a different package can only override the non-final methods declared public or protected.

An overriding method can throw any uncheck exceptions, regardless of whether the overridden method throws exceptions or not. However, the overriding method should not throw checked exceptions that are new or broader than the ones declared by the overridden method. The overriding method can throw narrower or fewer exceptions than the overridden method.

Constructors cannot be overridden.

Using the super Keyword

When invoking a superclass version of an overridden method thesuperkeyword is used.

Example

```
classAnimal{
publicvoid move (){
System.out.println ("Animals can move");
}
}
classDogextendsAnimal{
publicvoid move (){
super.move ();// invokes the super class method
System.out.println ("Dogs can walk and run");
}
}
publicclassTestDog{
publicstaticvoid main (String args []){
Animal b =newDog ();// Animal reference but Dog object
b.move ();// runs the method in Dog class
}
}
```

This will produce the following result

Output

```
Animals can move
Dogs can walk and run
```

Java Polymorphism

Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object.

Any Java object that can pass more than one IS-A test is considered to be polymorphic. In Java, all Java objects are polymorphic since any object will pass the IS-A test for their own type and for the class Object.

It is important to know that the only possible way to access an object is through a reference variable. A reference variable can be of only one type. Once declared, the type of a reference variable cannot be changed.

The reference variable can be reassigned to other objects provided that it is not declared final. The type of the reference variable would determine the methods that it can invoke on the object.

A reference variable can refer to any object of its declared type or any subtype of its declared type. A reference variable can be declared as a class or interface type.

Example

Let us look at an example.

```
publicinterfaceVegetarian{}
publicclassAnimal{}
publicclassDeerextendsAnimalimplementsVegetarian{}
```

Now, the Deer class is considered to be polymorphic since this has multiple inheritance. Following are true for the above examples

A Deer IS-A Animal

A Deer IS-A Vegetarian

A Deer IS-A Deer

A Deer IS-A Object

When we apply the reference variable facts to a Deer object reference, the following declarations are legal

Example

```
Deer d =newDeer ();
Animal a = d;
Vegetarian v = d;
Object o = d;
```

All the reference variables d, a, v, o refer to the same Deer object in the heap.

Virtual Methods

In this section, I will show you how the behavior of overridden methods in Java allows you to take advantage of polymorphism when designing your classes.

We already have discussed method overriding, where a child class can override a method in its parent. An overridden method is essentially hidden in the parent class, and is not invoked unless the child class uses the super keyword within the overriding method.

Example

```
/* File name: Employee.java */
publicclassEmployee{
privateString name;
privateString address;
privateint number;
publicEmployee (String name,String address,int number){
System.out.println ("Constructing an Employee");
this.name = name;
this.address = address;
this.number = number;
}
publicvoid mailCheck (){
System.out.println ("Mailing a check to "+this.name +""+this.address);
}
publicString toString (){
return name +""+ address +""+ number;
}
publicString getName (){
return name;
}
publicString getAddress (){
return address;
}
publicvoid setAddress (String newAddress){
address = newAddress;
}
publicint getNumber (){
return number;
}
}
```

Now suppose we extend Employee class as follows

```
/* File name: Salary.java */
publicclassSalaryextendsEmployee{
privatedouble salary;// Annual salary
publicSalary (String name,String address,int number,double salary){
super (name, address, number);
setSalary (salary);
}
publicvoid mailCheck (){
System.out.println ("Within mailCheck of Salary class ");
System.out.println ("Mailing check to "+ getName ()
+" with salary "+ salary);
}
publicdouble getSalary (){
return salary;
}
publicvoid setSalary (double newSalary){
if (newSalary >=0.0){
salary = newSalary;
}
}
publicdouble computePay (){
System.out.println ("Computing salary pay for "+ getName ());
return salary/52;
}
}
```

Now, you study the following program carefully and try to determine its output

```
/* File name: VirtualDemo.java */
publicclassVirtualDemo{
publicstaticvoid main (String [] args){
Salary s =newSalary ("Mohd Mohtashim","Ambehta, UP",3,3600.00);
Employee e =newSalary ("John Adams","Boston, MA",2,2400.00);
System.out.println ("Call mailCheck using Salary reference --");
s.mailCheck ();
System.out.println ("\n Call mailCheck using Employee reference--");
e.mailCheck ();
}
}
```

This will produce the following result

Output

```
Constructing an Employee
Constructing an Employee
Call mailCheck using Salary reference --
Within mailCheck of Salary class
ailing check to Mohd Mohtashim with salary 3600.0
Call mailCheck using Employee reference--
Within mailCheck of Salary class
ailing check to John Adams with salary 2400.0
```

Here, we instantiate two Salary objects. One using a Salary references, and the other using an Employee referencee.

While invokings.mailCheck (), the compiler sees mailCheck () in the Salary class at compile time, and the JVM invokes mailCheck () in the Salary class at run time.

mailCheck () oneis quite different becauseeis an Employee reference. When the compiler seese.mailCheck (), the compiler sees the mailCheck () method in the Employee class.

Here, at compile time, the compiler used mailCheck () in Employee to validate this statement. At run time, however, the JVM invokes mailCheck () in the Salary class.

This behavior is referred to as virtual method invocation, and these methods are referred to as virtual methods. An overridden method is invoked at run time, no matter what data type the reference is that was used in the source code at compile time.

Java Abstraction

As per dictionary,abstractionis the quality of dealing with ideas rather than events. For example, when you consider the case of e-mail, complex details such as what happens as soon as you send an e-mail, the protocol your e-mail server uses are hidden from the user. Therefore, to send an e-mail you just need to type the content, mention the address of the receiver, and click send.

Likewise in Object-oriented programming, abstraction is a process of hiding the implementation details from the user, only the functionality will be provided to the user. In other words, the user will have the information on what the object does instead of how it does it.

In Java, abstraction is achieved using Abstract classes and interfaces.

Abstract Class

A class which contains theabstractkeyword in its declaration is known as abstract class.

Abstract classes may or may not containabstract methods, i.e., methods without body (public void get ();)

But, if a class has at least one abstract method, then the classmustbe declared abstract.

If a class is declared abstract, it cannot be instantiated.

To use an abstract class, you have to inherit it from another class, provide implementations to the abstract methods in it.

If you inherit an abstract class, you have to provide implementations to all the abstract methods in it.

Example

This section provides you an example of the abstract class. To create an abstract class, just use theabstractkeyword before the class keyword, in the class declaration.

```
/* File name: Employee.java */
publicabstractclassEmployee{
privateString name;
privateString address;
privateint number;
publicEmployee (String name,String address,int number){
System.out.println ("Constructing an Employee");
this.name = name;
this.address = address;
this.number = number;
}
publicdouble computePay (){
System.out.println ("Inside Employee computePay");
return0.0;
}
publicvoid mailCheck (){
System.out.println ("Mailing a check to "+this.name +""+this.address);
}
publicString toString (){
return name +""+ address +""+ number;
}
publicString getName (){
return name;
}
publicString getAddress (){
return address;
}
publicvoid setAddress (String newAddress){
address = newAddress;
}
publicint getNumber (){
return number;
}
}
```

You can observe that except abstract methods the Employee class is same as normal class in Java. The class is now abstract, but it still has three fields, seven methods, and one constructor.

Now you can try to instantiate the Employee class in the following way

```
/* File name: AbstractDemo.java */
publicclassAbstractDemo{
publicstaticvoid main (String [] args){
/* Following is not allowed and would raise error */
Employee e =newEmployee ("George W.","Houston, TX",43);
System.out.println ("\n Call mailCheck using Employee reference--");
e.mailCheck ();
}
}
```

When you compile the above class, it gives you the following error

```
Employee.java:46: Employee is abstract; cannot be instantiated
Employee e = new Employee ("George W.", "Houston, TX", 43);
^
1 error
```

Inheriting the Abstract Class

We can inherit the properties of Employee class just like concrete class in the following way

Example

```
/* File name: Salary.java */
publicclassSalaryextendsEmployee{
privatedouble salary;// Annual salary
publicSalary (String name,String address,int number,double salary){
super (name, address, number);
setSalary (salary);
}
publicvoid mailCheck (){
System.out.println ("Within mailCheck of Salary class ");
System.out.println ("Mailing check to "+ getName ()+" with salary "+ salary);
}
publicdouble getSalary (){
return salary;
}
publicvoid setSalary (double newSalary){
if (newSalary >=0.0){
salary = newSalary;
}
}
publicdouble computePay (){
System.out.println ("Computing salary pay for "+ getName ());
return salary/52;
}
}
```

Here, you cannot instantiate the Employee class, but you can instantiate the Salary Class, and using this instance you can access all the three fields and seven methods of Employee class as shown below.

```
/* File name: AbstractDemo.java */
publicclassAbstractDemo{
publicstaticvoid main (String [] args){
Salary s =newSalary ("Mohd Mohtashim","Ambehta, UP",3,3600.00);
Employee e =newSalary ("John Adams","Boston, MA",2,2400.00);
System.out.println ("Call mailCheck using Salary reference --");
s.mailCheck ();
System.out.println ("\n Call mailCheck using Employee reference--");
e.mailCheck ();
}
}
```

This produces the following result

Output

```
Constructing an Employee
Constructing an Employee
Call mailCheck using Salary reference --
Within mailCheck of Salary class
Mailing check to Mohd Mohtashim with salary 3600.0
Call mailCheck using Employee reference--
Within mailCheck of Salary class
Mailing check to John Adams with salary 2400.0
```

Abstract Methods

If you want a class to contain a particular method but you want the actual implementation of that method to be determined by child classes, you can declare the method in the parent class as an abstract.

abstractkeyword is used to declare the method as abstract.

You have to place theabstractkeyword before the method name in the method declaration.

An abstract method contains a method signature, but no method body.

Instead of curly braces, an abstract method will have a semoi colon (;) at the end.

Following is an example of the abstract method.

Example

```
publicabstractclassEmployee{
privateString name;
privateString address;
privateint number;
publicabstractdouble computePay ();
// Remainder of class definition
}
```

Declaring a method as abstract has two consequences

The class containing it must be declared as abstract.

Any class inheriting the current class must either override the abstract method or declare itself as abstract.

Note Eventually, a descendant class has to implement the abstract method; otherwise, you would have a hierarchy of abstract classes that cannot be instantiated.

Suppose Salary class inherits the Employee class, then it should implement thecomputePay ()method as shown below

```
/* File name: Salary.java */
publicclassSalaryextendsEmployee{
privatedouble salary;// Annual salary
publicdouble computePay (){
System.out.println ("Computing salary pay for "+ getName ());
return salary/52;
}
// Remainder of class definition
}
```

Java - Encapsulation

Encapsulationis one of the four fundamental OOP concepts. The other three are inheritance, polymorphism, and abstraction.

Encapsulation in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class. Therefore, it is also known asdata hiding.

To achieve encapsulation in Java

Declare the variables of a class as private.

Provide public setter and getter methods to modify and view the variables values.

Example

Following is an example that demonstrates how to achieve Encapsulation in Java

```
/* File name: EncapTest.java */
publicclassEncapTest{
privateString name;
privateString idNum;
privateint age;
publicint getAge (){
return age;
}
publicString getName (){
return name;
}
publicString getIdNum (){
return idNum;
}
publicvoid setAge (int newAge){
age = newAge;
}
publicvoid setName (String newName){
name = newName;
}
publicvoid setIdNum (String newId){
idNum = newId;
}
}
```

The public setXXX () and getXXX () methods are the access points of the instance variables of the EncapTest class. Normally, these methods are referred as getters and setters. Therefore, any class that wants to access the variables should access them through these getters and setters.

The variables of the EncapTest class can be accessed using the following program

```
/* File name: RunEncap.java */
publicclassRunEncap{
publicstaticvoid main (String args []){
EncapTest encap =newEncapTest ();
encap.setName ("James");
encap.setAge (20);
encap.setIdNum ("12343ms");
System.out.print ("Name: "+ encap.getName ()+"
Age: "+ encap.getAge ());
}
}
```

This will produce the following result

Output

```
Name: James Age: 20
```

Benefits of Encapsulation

The fields of a class can be made read-only or write-only.

A class can have total control over what is stored in its fields.

Java Interfaces

An interface is a reference type in Java. It is similar to class. It is a collection of abstract methods. A class implements an interface, thereby inheriting the abstract methods of the interface.

Along with abstract methods, an interface may also contain constants, default methods, static methods, and nested types. Method bodies exist only for default methods and static methods.

Writing an interface is similar to writing a class. But a class describes the attributes and behaviors of an object. And an interface contains behaviors that a class implements.

Unless the class that implements the interface is abstract, all the methods of the interface need to be defined in the class.

An interface is similar to a class in the following ways

An interface can contain any number of methods.

An interface is written in a file with a.javaextension, with the name of the interface matching the name of the file.

The byte code of an interface appears in a.classfile.

Interfaces appear in packages, and their corresponding bytecode file must be in a directory structure that matches the package name.

However, an interface is different from a class in several ways, including

You cannot instantiate an interface.

An interface does not contain any constructors.

All of the methods in an interface are abstract.

An interface cannot contain instance fields. The only fields that can appear in an interface must be declared both static and final.

An interface is not extended by a class; it is implemented by a class.

An interface can extend multiple interfaces.

Declaring Interfaces

Theinterfacekeyword is used to declare an interface. Here is a simple example to declare an interface

Example

Following is an example of an interface

```
/* File name: NameOfInterface.java */
import java.lang.*;
// Any number of import statements
publicinterfaceNameOfInterface{
// Any number of final, static fields
// Any number of abstract method declarations\
}
```

Interfaces have the following properties

An interface is implicitly abstract. You do not need to use theabstractkeyword while declaring an interface.

Each method in an interface is also implicitly abstract, so the abstract keyword is not needed.

Methods in an interface are implicitly public.

Example

```
/* File name: Animal.java */
interfaceAnimal{
publicvoid eat ();
publicvoid travel ();
}
```

Implementing Interfaces

When a class implements an interface, you can think of the class as signing a contract, agreeing to perform the specific behaviors of the interface. If a class does not perform all the behaviors of the interface, the class must declare itself as abstract.

A class uses theimplementskeyword to implement an interface. The implements keyword appears in the class declaration following the extends portion of the declaration.

Example

```
/* File name: MammalInt.java */
publicclassMammalIntimplementsAnimal{
publicvoid eat (){
System.out.println ("Mammal eats");
}
publicvoid travel (){
System.out.println ("Mammal travels");
}
publicint noOfLegs (){
return0;
}
publicstaticvoid main (String args []){
MammalInt m =newMammalInt ();
m.eat ();
m.travel ();
}
}
```

This will produce the following result

Output

```
Mammal eats
Mammal travels
```

When overriding methods defined in interfaces, there are several rules to be followed

Checked exceptions should not be declared on implementation methods other than the ones declared by the interface method or subclasses of those declared by the interface method.

The signature of the interface method and the same return type or subtype should be maintained when overriding the methods.

An implementation class itself can be abstract and if so, interface methods need not be implemented.

When implementation interfaces, there are several rules

A class can implement more than one interface at a time.

A class can extend only one class, but implement many interfaces.

An interface can extend another interface, in a similar way as a class can extend another class.

Extending Interfaces

An interface can extend another interface in the same way that a class can extend another class. Theextendskeyword is used to extend an interface, and the child interface inherits the methods of the parent interface.

The following Sports interface is extended by Hockey and Football interfaces.

Example

```
// Filename: Sports.java
publicinterfaceSports{
publicvoid setHomeTeam (String name);
publicvoid setVisitingTeam (String name);
}
// Filename: Football.java
publicinterfaceFootballextendsSports{
publicvoid homeTeamScored (int points);
publicvoid visitingTeamScored (int points);
publicvoid endOfQuarter (int quarter);
}
// Filename: Hockey.java
publicinterfaceHockeyextendsSports{
publicvoid homeGoalScored ();
publicvoid visitingGoalScored ();
publicvoid endOfPeriod (int period);
publicvoid overtimePeriod (int ot);
}
```

The Hockey interface has four methods, but it inherits two from Sports; thus, a class that implements Hockey needs to implement all six methods. Similarly, a class that implements Football needs to define the three methods from Football and the two methods from Sports.

Extending Multiple Interfaces

A Java class can only extend one parent class. Multiple inheritance is not allowed. Interfaces are not classes, however, and an interface can extend more than one parent interface.

The extends keyword is used once, and the parent interfaces are declared in a comma-separated list.

For example, if the Hockey interface extended both Sports and Event, it would be declared as

Example

```
publicinterfaceHockeyextendsSports,Event
```

Tagging Interfaces

The most common use of extending interfaces occurs when the parent interface does not contain any methods. For example, the MouseListener interface in the java.awt.event package extended java.util.EventListener, which is defined as

Example

```
package java.util;
publicinterfaceEventListener
{}
```

An interface with no methods in it is referred to as ataggingintereface. There are two basic design purposes of tagging interfaces

Creates a common parent As with the EventListener interface, which is extended by dozens of other interfaces in the Java API, you can use a tagging interface to create a common parent among a group of interfaces. For example, when an interface extends EventListener, the JVM knows that this particular interface is going to be used in an event delegation scenario.

Adds a data type to a class This situation is where the term, tagging comes from. A class that implements a tagging interface does not need to define any methods (since the interface does not have any), but the class becomes an interface type through polymorphism.

Java - Packages

Packages are used in Java in order to prevent naming conflicts, to control access, to make searching/locating and usage of classes, interfaces, enumerations and annotations easier, etc.

APackagecan be defined as a grouping of related types (classes, interfaces, enumerations and annotations) providing access protection and namespace management.

Some of the existing packages in Java are

java.lang bundles the fundamental classes

java.io classes for input, output functions are bundled in this package

Programmers can define their own packages to bundle group of classes/interfaces, etc. It is a good practice to group related classes implemented by you so that a programmer can easily determine that the classes, interfaces, enumerations, and annotations are related.

Since the package creates a new namespace there won't be any name conflicts with names in other packages. Using packages, it is easier to provide access control and it is also easier to locate the related classes.

Creating a Package

While creating a package, you should choose a name for the package and include apackagestatement along with that name at the top of every source file that contains the classes, interfaces, enumerations, and annotation types that you want to include in the package.

The package statement should be the first line in the source file. There can be only one package statement in each source file, and it applies to all types in the file.

If a package statement is not used then the class, interfaces, enumerations, and annotation types will be placed in the current default package.

To compile the Java programs with package statements, you have to use -d option as shown below.

```
javac -d Destination_folder file_name.java
```

Then a folder with the given package name is created in the specified destination, and the compiled class files will be placed in that folder.

Example

Let us look at an example that creates a package calledanimals. It is a good practice to use names of packages with lower case letters to avoid any conflicts with the names of classes and interfaces.

Following package example contains interface namedanimals

```
/* File name: Animal.java */
package animals;
interfaceAnimal{
publicvoid eat ();
publicvoid travel ();
}
```

Now, let us implement the above interface in the same packageanimals

```
package animals;
/* File name: MammalInt.java */
publicclassMammalIntimplementsAnimal{
publicvoid eat (){
System.out.println ("Mammal eats");
}
publicvoid travel (){
System.out.println ("Mammal travels");
}
publicint noOfLegs (){
return0;
}
publicstaticvoid main (String args []){
MammalInt m =newMammalInt ();
m.eat ();
m.travel ();
}
}
```

Now compile the java files as shown below

```
$ javac -d. Animal.java
$ javac -d. MammalInt.java
```

Now a package/folder with the nameanimalswill be created in the current directory and these class files will be placed in it as shown below.

You can execute the class file within the package and get the result as shown below.

```
Mammal eats
Mammal travels
```

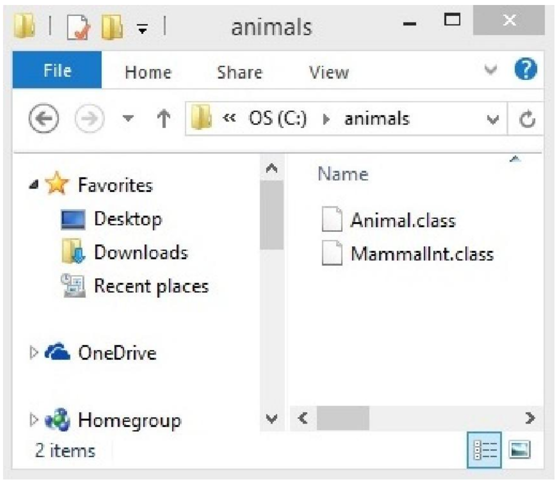
The import Keyword

If a class wants to use another class in the same package, the package name need not be used. Classes in the same package find each other without any special syntax.

Example

Here, a class named Boss is added to the payroll package that already contains Employee. The Boss can then refer to the Employee class without using the payroll prefix, as demonstrated by the following Boss class.

```
package payroll;
publicclassBoss{
publicvoid payEmployee (Employee e){
e.mailCheck ();
}
}
```



Animal.class
MammalInt.class

What happens if the Employee class is not in the payroll package? The Boss class must then use one of the following techniques for referring to a class in a different package.

The fully qualified name of the class can be used. For example

payroll.Employee

The package can be imported using the import keyword and the wild card (*). For example

import payroll.*;

The class itself can be imported using the import keyword. For example

import payroll.Employee;

Note A class file can contain any number of import statements. The import statements must appear after the package statement and before the class declaration.

The Directory Structure of Packages

Two major results occur when a class is placed in a package

The name of the package becomes a part of the name of the class, as we just discussed in the previous section.

The name of the package must match the directory structure where the corresponding bytecode resides.

Here is simple way of managing your files in Java

Put the source code for a class, interface, enumeration, or annotation type in a text file whose name is the simple name of the type and whose extension is.java.

For example

```
// File Name: Car.java
package vehicle;
publicclassCar{
// Class implementation.
}
```

Now, put the source file in a directory whose name reflects the name of the package to which the class belongs

....\vehicle\Car.java

Now, the qualified class name and pathname would be as follows

Class name vehicle.Car

Path name vehicle\Car.java (in windows)

In general, a company uses its reversed Internet domain name for its package names.

Example A company's Internet domain name is apple.com, then all its package names would start with com.apple. Each component of the package name corresponds to a subdirectory.

Example The company had a com.apple.computers package that contained a Dell.java source file, it would be contained in a series of subdirectories like this

....\com\apple\computers\Dell.java

At the time of compilation, the compiler creates a different output file for each class, interface and enumeration defined in it. The base name of the output file is the name of the type, and its extension is.class.

For example

```
// File Name: Dell.java
package com.apple.computers;
publicclassDell{
}
classUps{
}
```

Now, compile this file as follows using -d option

```
$javac -d. Dell.java
```

The files will be compiled as follows

```
.\com\apple\computers\Dell.class
.\com\apple\computers\Ups.class
```

You can import all the classes or interfaces defined in\com\apple\computers\as follows

import com.apple.computers.*;

Like the.java source files, the compiled.class files should be in a series of directories that reflect the package name. However, the path to the.class files does not have to be the same as the path to the.java source files. You can arrange your source and class directories separately, as

<path-one>\sources\com\apple\computers\Dell.java

<path-two>\classes\com\apple\computers\Dell.class

By doing this, it is possible to give access to the classes directory to other programmers without revealing your sources. You also need to manage source and class files in this manner so that the compiler and the Java Virtual Machine (JVM) can find all the types your program uses.

The full path to the classes directory, <path-two>\classes, is called the class path, and is set with the CLASSPATH system variable. Both the compiler and the JVM construct the path to your.class files by adding the package name to the class path.

Say <path-two>\classes is the class path, and the package name is com.apple.computers, then the compiler and JVM will look for.class files in <path-two>\classes\com\apple\computers.

A class path may include several paths. Multiple paths should be separated by a semicolon (Windows) or colon (Unix). By default, the compiler and the JVM search the current directory and the JAR file containing the Java platform classes so that these directories are automatically in the class path.

Set CLASSPATH System Variable

To display the current CLASSPATH variable, use the following commands in Windows and UNIX (Bourne shell)

In Windows C:\> set CLASSPATH

In UNIX % echo $CLASSPATH

To delete the current contents of the CLASSPATH variable, use

In Windows C:\> set CLASSPATH =

In UNIX % unset CLASSPATH; export CLASSPATH

To set the CLASSPATH variable

In Windows set CLASSPATH = C:\users\jack\java\classes

In UNIX % CLASSPATH = /home/jack/java/classes; export CLASSPATH

Note: You can dig deeper from here if you want to know more about Java.

https://www.tutorialspoint.com/java/java_data_structures.htm

Structured Query Language

SQL: Structured Query Language -- a computer language developed by American computer scientists Donald D. Chamberlin and Raymond F. Boyce at IBM in 1974 to create database, store, manipulate, delete and retrieve data stored in database.

How to create database in MySQL

First you have to open MySQL terminal and then you have to enter the command:

create database data;

or

CREATE DATABASE data;

And press enter. Then

Query OK, 1 row affected (0.01 sec)

will be displayed on the console screen indicating that database named data is created. And if you enter the command:

show databases;

And press enter. Then

Database

CodingGround

data

information_schema

mysql

performance_schema

will be displayed on the console screen. And if you want to create a table in the database "data", then you have to enter the command:

use data;

And press enter. Then

Database changed

will be displayed on the console screen stating that your active database is now "data". And if you want to create a table named "states" with three fields: id, state, and population:

id	state	population

in your active database named "data", then you have to enter the command:

CREATE TABLE states (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, state CHAR (25), population INT (9));

And press enter. Then

Query OK, 0 rows affected (0.07 sec)

will be displayed on the console screen stating that the above table is created.

Note:

The INT command will make the id field contain only numbers (i.e., integers).

The NOT NULL command makes sure that the id field cannot be left blank / empty.

The PRIMARY KEY designates the id field as the key field in the table.

The AUTO_INCREMENT command will automatically assign increasing values into the id field, essentially automatically numbering each entry.

The CHAR (characters) and INT (integers) commands designate the types of data allowed in those fields. The number next to the commands CHAR and INT indicate how many characters or integers can fit in the field.

Now it's time to start entering your information. Use the following command:

```
INSERT INTO states (id, state, population) VALUES (NULL, 'Karnataka', 256666);
INSERT INTO states (id, state, population) VALUES (NULL, 'Assam', 2568585);
INSERT INTO states (id, state, population) VALUES (NULL, 'Kashmir', 2569);
```

to input your entry. Then

```
Query OK, 1 row affected (0.03 sec)
Query OK, 1 row affected (0.01 sec)
Query OK, 1 row affected (0.00 sec)
```

will be displayed on the console screen stating that you have inputted your entry. And if you enter the following command:

```
select*from states;
```

Then, your created table named "states" will be displayed on the screen as follows:

id	state	population
1	Karnataka	256666
2	Assam	2568585
3	Kashmir	2569

And if you wish to create the following table:

id	state	population	language
1	Karnataka	256666	Kannada
2	Assam	2569	Assami

You have to use the following command:

```
CREATE TABLE states (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, state CHAR (25), population INT (9), language CHAR (25));
```

And press enter and

```
Query OK, 0 rows affected (0.03 sec)
```

will be displayed on the console screen and then you should enter the following command:

```
INSERT INTO states (id, state, population, language) VALUES (NULL, 'Karnataka', 256666, 'Kannada');
INSERT INTO states (id, state, population, language) VALUES (NULL,'Assam',2569,'Assami');
```

And press enter and

```
Query OK, 1 row affected (0.01 sec)
Query OK, 1 row affected (0.00 sec)
```

will be displayed on the console screen and if you enter the command:

```
select*from states;
```

Then the above table will be displayed on the screen.

If you enter the command:

```
select state, population from states;
```

Then

state	population
Karnataka	256666
Assam	2569

will be displayed on the console screen. And if you enter the command:

```
select state from states;
```

Then

state
Karnataka
Assam

will be displayed on the console screen.

If you enter the command:

```
select*from states where language ='kannada';
```

Then

id	state	population	language
1	Karnataka	256666	Kannada

will be displayed on the console screen. Similarly, if you enter the command:

```
select * from states where id =2;
```

Then

id	state	population	language
2	Assam	2569	Assami

will be displayed on the console screen.

SQL and & or Command:

If you enter the command:

```
select*from states where population =256666 and language ='kannada';
```

or

```
select*from states where population = 226666 or language = 'kannada';
```

Then

id	state	population	language
1	Karnataka	256666	Kannada

will be displayed on the console screen.

If you enter the command:

```
select *from states where population = 256666 or language ='assami';
```

id	state	population	language
1	Karnataka	256666	Kannada
2	Assam	2569	Assami

How to insert information into the table

If you enter the command:

INSERT INTO states (id, state, population, language) VALUES (NULL, 'tamil nadu', 288,'tamil');

Then

id	state	population	language
1	Karnataka	256666	Kannada
2	Assam	2569	Assami
3	tamil nadu	288	tamil

will be displayed on the console screen.

UPDATE INFORMATION:

If you enter the command:

update states set language = 'telagu', id = 1 where state = 'Karnataka';

Then

Query OK, 1 row affected (0.01 sec)

Rows matched: 1 Changed: 1 Warnings: 0

will be displayed on the console screen. And if you enter the command:

select * from states;

Then

id	state	population	language
1	Karnataka	256666	telagu
2	Assam	2569	Assami
3	tamil nadu	288	tamil

will be displayed on the console screen.

DELETE information:

If you enter the command:

delete from states where language ='assami' and state ='assam';

Then

Query OK, 1 row affected (0.00 sec)

will be displayed on the console screen. And if you enter the command:

select * from states;

Then

id	state	population	language
1	Karnataka	256666	telagu
3	tamil nadu	288	tamil

will be displayed on the console screen.

How to delete database in MySQL:

Note: If want to delete database "dbtest" from MySQL. Then you have to enter the command:

drop database dbtest;

Then

Query OK, 1 row affected (0.00 sec)

will be displayed on the console screen stating that database "dbtest" is deleted from MySQL.

If want to delete table "states" from database "dbtest." Then you have to enter the command:

drop table states;

Then

Query OK, 1 row affected (0.00 sec)

will be displayed on the console screen stating that table "states" is deleted from database "dbtest".

Limit Data Selection From MySQL Database:

If enter the command:

select*from states limit 1;

Then

id	state	population	language
1	Karnataka	256666	Kannada

will be displayed on the console screen.

If enter the command:

select*from states limit 2;

Then

id	state	population	language
1	Karnataka	256666	Kannada
2	Assam	2569	Assami

will be displayed on the console screen. If you enter the command: truncate states; Then Query OK, 0 rows affected (0.06 sec) will be displayed on the console screen stating that all the rows are removed from the table "states". And you can confirm it by entering the command: select*from states; Then: Empty set (0.01 sec) will be displayed on the console screen. Python Python is a popular, very powerful high-level language (like C, C++, Perl, and Java and its name is derived from "Monty Python's Flying Circus" a British television series), object-oriented programming scripting language designed by Dutch programmer "Guido van Rossum" in the early 1990s (often referred to as a "glue" language, meaning that it is capable to work in mixed-language environment) which is easy to understand, easy to use, write, modify and debug and flexible and easy to implement and run on open source operating systems like Linux, Windows, Macintosh, Solaris, FreeBSD, OS/2, Amiga, AROS, AS/400 and is employed to perform automated testing of applications (i.e., to execute tests of applications, report outcomes and compare results with earlier test runs) and to increase the effectiveness and speed of software testing and its other commercial uses include financial applications, educational software, games, production of special effects for such movies as The Phantom Menace and The Mummy Returns, and business software. And python might not be the best choice for building the following types of applications and systems: Graphics-intensive applications, such as action games -- where performance is important (because it is true that CODE WRITTEN IN Python, c# and visual basic etc. is far slower than the same code write in C++. Hence, C++ is necessarily preferred). And its support for Matlab-like array manipulation and plotting is a major reason to prefer it over other high level programming languages such as Perl and Ruby. You should be very careful while working on python code. Indentation in python takes the center-stage. You cannot write a loop or a conditional statement without using indentation. There is a popular saying that code in python should be as guido indented it not how he intended it. A simple python program to print the word "Hello World!" on screen: # Hello World program in Python print"Hello World!\n" Output on the screen: Hello World! print "Hello World!\n" implies the statement that make provision to print: Hello World! on the screen. # Hello World program in Python the statement that implies: comment What will be the output on the screen: (a) print "Hello World!\n" print "Hello World!\n" Answer: Hello World! Hello World! (b) print "Hello World!" print "Hello World!" Answer: Hello World! Hello World! What is the mistake in the following program: Hello World program in Python print "Hello World!\n" Answer: # is not added to the statement: Hello World program in Python

```
//-----------------------------------------------------------------
-----------------------------------------------------------
#Hello World program in Python print "Hello World!\n"
-----------------------------------------------------------------
-----------------------------------------------------------//
```

Program 1.1 Python program to add two numbers: number1=2 number2=3 print "The sum of {0} and {1} is {2}".format (number1, number2, float (number1) + float (number2)) Output on the screen: The sum of 2 and 3 is 5.0 print "The sum of {0} and {1} is {2}".format (number1, number2, float (number1) + float (number2)) implies the statement that make provision to print the output: The sum of 2 and 3 is 5.0 on the screen. Note: If you replace the statement: print "The sum of {0} and {1} is {2}".format (number1, number2, float (number1) + float (number2)) by the statement: print "The sum of {1} and {2} is {3}".format (number1, number2, float (number1) + float (number2)) Then Runtime error or IndexError: tuple index out of range will be displayed on the screen. Note: Dot notation (.) in the statement: print "The sum of {0} and {1} is {2}".format (number1, number2, float (number1) + float (number2)) connects the two statements: "The sum of {0} and {1} is {2}" format (number1, number2, float (number1) + float (number2)) And if you forget to write dot notation in the statement: print "The difference of {0} and {1} is {2}".format (number1, number2, float (number1) - float (number2)) i.e., print "The difference of {0} and {1} is {2}" format (number1, number2, float (number1) - float (number2)) Then Syntax error will be displayed on the console screen. In the statement: print "The sum of {0} and {1} is {2}".format (number1, number2, float (number1) + float (number2)) {0} act as a placeholder for number1 in the format method {1} act as a placeholder for number2 in the format method {2} act as a placeholder for float (number1) + float (number2) in the format method If you want to enter the values for number1 and number2 through keyboard, then you need to replace the statements: number1 = 2 number2 = 3 by the statements: number1=input (" Please Enter the First Number: ") number2=input (" Please Enter the Second Number: ") i.e., the above program should be rewritten as: number1=input (" Please Enter the First Number: ") number2=input (" Please Enter the Second Number: ") print "The sum of {0} and {1} is {2}".format (number1, number2, float (number1) + float (number2)) Output on the screen: Please Enter the First Number: If you enter 1 Please Enter the Second Number: If you enter 2 The sum of 1 and 2 is 3.0 will be displayed on the screen. Statements: number1 = input (" Please Enter the First Number: ") number2 = input (" Please Enter the Second Number: ") ask the user to enter two integer numbers and stores the user entered values in variables number 1 and number 2. Program 1.2 Python program to subtract two numbers: number1=input (" Please Enter the First Number: ") number2=input (" Please Enter the Second Number: ") print "The difference of {0} and {1} is {2}".format (number1, number2, float (number1) - float (number2)) Output on the screen: Please Enter the First Number: If you enter 6 Please Enter the Second Number: If you enter 4 The difference of 6 and 4 is 2.0 will be displayed on the screen. Program 1.3 Python program to divide two numbers: number1=input (" Please Enter the First Number: ") number2=input (" Please Enter the Second Number: ") print "The division of {0} by {1} yields {2}".format (number1, number2, float (number1) / float (number2)) Output on the screen: Please Enter the First Number: If you enter 6 Please Enter the Second Number: If you enter 2 The division of 6 by 2 is 3.0 will be displayed on the screen. Program 1.4 Python program to multiply two numbers: number1=input (" Please Enter the First Number: ") number2=input (" Please Enter the Second Number: ") print "The product of {0} and {1} is {2}".format (number1, number2, float (number1) * float (number2)) Output on the screen: Please Enter the First Number: If you enter 6 Please Enter the Second Number: If you enter 2 The product of 6 and 2 is 12.0 will be displayed on the screen. Program 1.5 Python program to find the area of a circle: number1=input (" Please Enter the radius: ") print "The area of the circle is {0}".format (4*3.14* number1* number1) Output on the screen: Please Enter the radius: If you enter 2 The area of the circle is 50.24 will be displayed on the screen. Program 1.6 Python program to find the square root of a number: number1=input (" Please Enter the number: ") print "The square root of the entered number is {0}".format (number1 ** 0.5) Output on the screen: Please Enter the number: If you enter 4 The square root of the entered number is 2.0 will be displayed on the screen. 1/2=0.5 ** implies: exponent operator number1 ** 0.5 implies: number1 ** 1/2 (which implies number1 to the power of 1/2) Program 1.7 Python program to find the square of a number number1 = input (" Please Enter the number: ") print "The square of the entered number is {0}".format (number1 * number1) Output on the screen: Please Enter the number: If you enter 4 The square of the entered number is 16.0 will be displayed on the screen. Program 1.8 Python program to find the incremented and decremented values of the entered number: number1 = input (" Please Enter the number: ") print "The increment of the entered number is {0}".format (number1 +1) print "The decrement of the entered number is {0}".format (number1 - 1) Output on the screen: Please Enter the number: If you enter 6 The increment of the entered number is 7 The decrement of the entered number is 5 will be displayed on the screen What will be the output of the following programs: (a) x = 13 if x < 10: print ("Good morning") elif x<12: print ("Soon time for lunch") elif x<18: print ("Good day") elif x<22: print ("Good evening") else: print ("Good night") Answer: Good day (b) n1 = [0 for i in range (15)] print (n1) Answer: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] If you replace the statement: n1 = [0 for i in range (15)] by the statement: n1 = ["computer" for i in range (15)] Then the output on the screen is: ['computer', 'computer', 'computer', 'computer', 'computer', 'computer', 'computer', 'computer', 'computer', 'computer', 'computer', 'computer', 'computer', 'computer', 'computer'] (c) for k in range (1,10): print (k) Answer: 1 2 3 4 5 6 7 8 9 If the statement: for k in range (1,10): by the statement: for k in range (0,10): Then the output on the screen is: 0 1 2 3 4 5 6 7 8 9 If the statement: print (k) is replaced by the statement: print ("computer") i.e., for k in range (0,10): print ("computer") Then the output on the screen is: computer computer computer computer computer computer computer computer computer computer (d) for day in ["Sunday","Monday","Tuesday","Wednesday","Thursday", "Friday","Saturday"]: print (day) Answer: Sunday Monday Tuesday Wednesday Thursday Friday Saturday (e) n = 10 sum = 0 for i in range (1,n):

sum = sum + i print sum Answer: 45 range (1,n) means: generate integers starting from 1 up to, but not including, n. How the execution takes place? i=1 (sum = 0 because the sum is initialized to 0 in the statement sum = 0) Is i in range (1,n) true? Yes, do this sum = sum + i = 0 +1 =1 Now i=2 (now the sum = 1) Is i in range (1,n) true? Yes, do this sum = sum + i = 1 +2 =3 Now i=3 (now the sum = 3) Is i in range (1,n) true? Yes, do this sum = sum + i = 3 +3 = 6 Now i=4 (now the sum = 6) Is i in range (1,n) true? Yes, do this sum = sum + i = 6 + 4= 10 Now i=5 (now the sum = 10) Is i in range (1,n) true? Yes, do this sum = sum + i = 10 + 5= 15 Now i=6 (now the sum = 15) Is i in range (1,n) true? Yes, do this sum = sum + i = 15 + 6 = 21 Now i=7 (now the sum = 21) Is i in range (1,n) true? Yes, do this sum = sum + i = 21 + 7 = 28 Now i=8 (now the sum = 28) Is i in range (1,n) true? Yes, do this sum = sum + i = 28 + 8 = 36 Now i=9 (now the sum = 36) Is i in range (1,n) true? Yes, do this sum = sum + i = 36 + 9 = 45 stops because range (1,n) means: generate integers starting from i=1 up to, but not including, i=10. If you want to enter the value for n through the keyboard, then the above program should take the form: n =input (" Please Enter the number: ") sum = 0 for i in range (1,n): sum = sum + i print sum Output on the screen: Please Enter the number: If you enter 10 Then 45 will be outputted on the console screen. (f) for k in range (1,11): print ("5 x {0} = {1}".format (k, 5*k)) Answer: 5 x 1 = 5 5 x 2 = 10 5 x 3 = 15 5 x 4 = 20 5 x 5 = 25 5 x 6 = 30 5 x 7 = 35 5 x 8 = 40 5 x 9 = 45 5 x 10 = 50 If you replace the statement: for k in range (1,11): by the statement: for k in range (0,11): Then the output on the screen is: 5 x 0 = 0 5 x 1 = 5 5 x 2 = 10 5 x 3 = 15 5 x 4 = 20 5 x 5 = 25 5 x 6 = 30 5 x 7 = 35 5 x 8 = 40 5 x 9 = 45 5 x 10 = 50 The Advanced Python Linux Introduction Every desktop computer uses an operating system. The most popular operating systems in use today are: Windows, Mac OS, and LINUX. Linux is the best-known notoriously reliable and highly secure open source portable operating system -- very much like UNIX -- that has become very popular over the last several years -- originally created as a labor of love by Linus Torvalds -- computer science student at the University of Helsinki in Finland -- in the early 1990s and later developed by more than a thousand people around the world. Linux is fast, free and easy to use, that sits underneath all of the other software on a computer -- runs your computer -- handling all interactions between you and the hardware i.e., whether you're typing a letter, calculating a money budget, or managing your food recipes on your computer, the Linux operating system (similar to other Operating Systems, such as Windows XP, Windows 7, Windows 8, and Mac OS X) provides the essential air that your computer breathes. Linux is the most important technology advancement of the twenty-first century and Licensed under the General Public License (GPL) that Linux uses ensures that the software will always be open to anyone and whose source code is open and available for any user to review, which makes it easier to identify and repair vulnerabilities and it power the laptops, development machines and servers at Google, Facebook, Twitter, NASA, and New York Stock Exchange, just to name a few. Linux has many more features to amaze its users such as: Live CD/USB, Graphical user interface (X Window System) etc. Why LINUX? Although Microsoft Windows (which is the most likely to be the victim of viruses and malware) has made great improvements in reliability in recent years, it is considered less reliable than Linux. Linux is notoriously reliable and secure and it is free from constant battling viruses and malware (which may affect your desktops, laptops, and servers by corrupting files, causing slow downs, crashes, costly repairs and taking over basic functions of your operating system) and it keep yourself free from licensing fees i.e., zero cost of entry... as in free. You can install Linux on as many reliable computer ecosystems on the planet as you like without paying a cent for software or server licensing. While Microsoft Windows usually costs between $99.00 and $199.00 USD for each licensed copy and fear of losing data. Below are some examples of where Linux is used today: Android phones and tablets Servers TV, Cameras, DVD players, etc. Amazon Google U.S. Postal service New York Stock Exchange Basic LINUX COMMANDS $date display date and time $cal display calendar $date & cal display date, time and calendar $cal 8 2016 display August month 2016 year calendar $clear clear the screen $exit exit and get to login page $free displays information about RAM and swap space usage, showing the total and the used amount in both categories $free -b displays the output in bytes $free -k displays the output in kilobytes $free -m displays the output in megabytes $passwd change password $uname display the name of the current operating system $echo "Hello World" print the word Hello World $hostname display the Kernel version $echo " my username is $USER " print the output: my username is root $shutdown shutdown the operating system $logname display the username $whoami display the username The commands:$logname and$whoami display the username $echo 'a =20; b =64; print (a+b)' | python print the output: 84 $echo 'a =56; b =2; print (a / b)' | python print the output: 28 $echo 'a =6; b =2; print (a * b)' | python print the output: 12 $echo 'a =6; b =2; print (a - b)' | python print the output: 4 $echo 'a =5; print (pow (a,2))' | python print the output: 25 $echo 'a =4; print (format (a**0.5)) ' | python print the output: 2.0 (find square root) $date -- set 1998-11-02 to set current date as 02 Nov 1988 $date -- set 12:11:02 to set current time as 12:11:02 IST $printf "hi computer" print the output: hi computer $du display the information of disk usage of files and directories $who display the user details $ifconfig view and change the configuration of the network interfaces on your system $pwd display the present working directory $uname -a display the system architecture information $hostname -i display the IP address of the hostname $ls display the list of files $ps display the process information $uptime display the system running information $netstat -a display all ports (both TCP and UDP) $netstat -at display only TCP (Transmission Control Protocol) port connections $netstat -au display only UDP (User Datagram Protocol) port connections $netstat -I display all active listening ports connections $netstat -It display all active listening TCP ports $netstat -lu display all active listening UDP ports $netstat -lx display all active UNIX listening ports $ifconfig display all the active interfaces details $ifconfig -a display information of all network interfaces $df displays information about the total disk space, the disk space currently in use, and the free space on all the mounted drives $df -H display the number of occupied blocks in gigabytes, megabytes, or kilobytes $du -a displays the size of each individual file $du -s displays only the calculated total size $top provides a quick overview of the currently running processes $shutdown -h now shutdown the system and turn the power off immediately $shutdown -h +10 shutdown the system after 10 minutes While lots of organizations, corporations, and individuals have already been convinced for 20 years now that this is the way to go -- private users, training companies, universities, research centers, and commercial vendors needed applications like the Internet to make them realize they can profit from Open Source. Now Linux (successfully being used by several millions of users worldwide) has grown past the stage where it was almost exclusively an academic system, useful only to a handful of people with a technical background. It provides more than the operating system: there is an entire infrastructure supporting the chain of effort of creating an operating system, of making and testing programs for it, of bringing everything to the users, of supplying maintenance, updates and support and customizations, runs on numerous different platforms including the Intel and Alpha platform. Today, Linux is ready to accept the challenge of a fast-changing world to do various types of operations, call application programs etc. PHP (Hypertext Preprocessor) PHP / Hypertext Preprocessor (designed by an Greenlandic-Danish programmer "Rasmus Lerdorf" in 1994 as an efficient alternative to other scripting languages like Ruby, Perl and Microsofts ASP) is an relatively free (not licensed by a major corporation) popular efficient server side programming language (and relatively easy one to master and quick to learn) that carries out common website duties like accepting passwords, authenticating users, and managing forum posts and guest books. A simple PHP program to print the word "Hello World!" on screen: <?php echo "Hello World!";?> In the above example: <?php and?>denote: opening and closing tags within which the execution of php codes takes place. echo "Hello World!"; denote: the statement that makes provision to print the output: Hello World! on the screen. Even If you replace the statement: echo "Hello World!"; by the statement: print "Hello World!"; i.e., <?php print "Hello World!";?> There will be no change in the output on the screen (i.e., echo and print are more or less the same. They are both used to output data to the console screen). If replace the opening tag <?php by <? Then the output on the screen is: <? print "Hello World!"; ?> i.e., the entire program will be reflected on the console screen. Even if you write the statement: print ("Hello World!"); instead of the statement: print "Hello World!"; There will be no change in the output on the screen. Note: Even if youomitthe closing tag?> in the above program. There will be no change in the output on the screen. But sometimes it reflects error. So omission of?>is discouraged. Program 1.1 a) <?php echo "Hello World!"; echo "Hello World!";?> Output on the screen: Hello World!Hello World! b) <?php echo "\n Hello World!"; echo "\n Hello World!";?> Output on the screen: Hello World! Hello World! c) <?php echo "Hello World!"; echo "\t Hello World!";?> Output on the screen: Hello World! Hello World! Program 1.2 PHP program to add two numbers: <?php $num1 =1; $num2=5; $sum = $num1 + $num2; echo "Sum of the two numbers is: $sum";?> Output on the screen: Sum of the two numbers is: 6 Equal sign implies: storage operator. The statements: $num1 =1; $num2=5; $sum = $num1 + $num2; imply: that we are creating the variables $num1, $num2 and $sum and storing the values for created variables (i.e., 1 for $num1, 5 for $num1 and $sum for $num1 + $num2). The statement: echo "Sum of the two numbers is: $sum"; make provision to print the output: Sum of the two numbers is: 6 (which is 1+5) on the screen. Note: Suppose if you omit the $ symbol before a variable name (whose purpose is to make it clear that the word following the symbol $ is a variable the symbol $ distinguishes variables from other things) in the above program i.e., if you rewrite the above program as: <?php num1 = 1; num2=5; sum = num1 + num2; echo "Sum of the two numbers is: sum";?> Then PHP Parse error: syntax error, unexpected '=' will be displayed on the console screen. If you want to supply the integer values for $num1 and $num2 through the key board, then the statements: $num1 =1; $num2=5; should be replaced by the statements: echo "Please enter the first number: "; fscanf (STDIN, %d\n, $num1); echo "Please enter the second number: "; fscanf (STDIN, %d\n, $num2); i.e., <?php echo "Please enter the first number: "; fscanf (STDIN, "%d\n", $num1); echo "Please enter the second number: "; fscanf (STDIN, "%d\n", $num2); $sum = $num1 + $num2; echo "Sum of the two numbers is: $sum";?> Output on the screen: Please enter the first number: If you enter 5 Please enter the second number: If you enter 6 Sum of the two numbers is: 11 will be outputted on the screen. fscanf (STDIN, "%d\n", $num1); denote: the statement that make provision to enter a integer value for $num1 through the keyboard. fscanf (STDIN, "%d\n", $num2); denote: the statement that make provision to enter a integer value for $num2 through the keyboard. Format string %d in the statement: fscanf (STDIN, "%d\n", $num1); or in the statement: fscanf (STDIN, "%d\n", $num2); tells the input function fscanf () to read the number entered through the keyboard (which is a integer) and STDIN (which stands for Standard input) means feed the number entered through the keyboard into the program. If you want to enter the floating values (say 1.6 & 1.9) for $num1 and $num2, then your program should take the form: <?php echo "Please enter the first number: "; fscanf (STDIN, "%f\n", $num1); echo "Please enter the second number: "; fscanf (STDIN, "%f\n", $num2); $sum = $num1 + $num2; echo "Sum of the two numbers is: $sum";?> Output on the screen: Please enter the first number: If you enter 1.6 Please enter the second number: If you enter 1.9 Sum of the two numbers is: 3.5 will be outputted on the screen. Program 1.3 PHP program to subtract two numbers: <?php $num1 =5; $num2=1; $sub = $num1 - $num2; echo "difference of the two numbers is: $sub";?> Output on the screen: difference of the two numbers is: 4 Program 1.4 PHP program to divide two numbers: <?php $num1 =6; $num2=2; $div = $num1 / $num2; echo "the division of two numbers is: $div";?> Output on the screen: the division of two numbers is: 3 Program 1.5 PHP program to multiply two numbers: <?php $num1 = 6; $num2 = 2; $mult = $num1 * $num2; echo "the product of two numbers is: $mult";?> Output on the screen: the product of two numbers is: 12 Program 1.6 PHP program to find the area of a circle: <?php $radius = 2.0; $pi = 3.14159; $area = $pi * $radius * $radius; echo ("\n radius = $radius centimeter"); echo ("\n area = $area centimeter square");?> Output on the screen: radius = 2 centimeter area = 12.56636 centimeter square Program 1.7 PHP program to find the square root of a number <?php $num1 = 4.0; $num2 = sqrt ($num1); echo ("The square root of a number = $num2");?> Output on the screen: The square root of a number = 2 Program 1.8 PHP program to find the square of a number <?php $num1 = 2.0; $num2 = $num1 * $num1; echo ("\n the square of a number = $num2");?> Output on the screen: the square of a number = 4 If the statement: $num2 = $num1 * $num1; is replaced by: $num2 = pow (($num1), 2); i.e., if the above program is rewritten as: <?php $num1 = 2.0; $num2 = pow (($num1), 2); echo ("\n the square of a number = $num2");?> Then there will be no change in the output on the screen i.e., the square of a number = 4 will be outputted on the screen. Which means: $num2 = pow (($num1), 2); is the same as $num2 = $num1 * $num1; Program 1.9 PHP program to find the cube root of a number <?php $num1 = 6.0; $num2 = pow (($num1), 1/3); echo ("\n the cube root of a number = $num2");?> Output on the screen: the cube root of a number = 1.8171205928321 Program 2.0 PHP program to round off a number <?php $num1 = 4.5; $num2 = round ($num1); echo ("\n the round off of a number = $num2");?> Output on the screen: the round off of a number = 5 Program 2.1 PHP program to find the incremented and decremented values of two numbers. <?php $num1 =2; $num2=3; $num3 = $num1 +1; $num4 = $num1 - 1; $num5 = $num2 +1; $num6 = $num2 - 1; echo ("\n The incremented value of $num1 = $num3 "); echo ("\n The decremented value of $num1 = $num4 "); echo ("\n The incremented value of $num2 = $num5 "); echo ("\n The decremented value of $num2 = $num6 ");?> Output on the screen: The incremented value of 2 = 3 The decremented value of 2 = 1 The incremented value of 3 = 4 The decremented value of 3 = 2 Program 2.2 PHP program to find the greatest of two numbers using if else statement The syntax of if else statement is: if (this condition is true) { print this statement; } else { print this statement; } <?php $x = 4.5; $y=5; if ($x>$y){ echo (" x is greater than y"); } else { echo (" y is greater than x"); }?> Output on the screen: y is greater than x Note: if the above program is rewritten as: <?php $x = 4.5; $y=5; if ($x>$y){ echo (" $x is greater than $y"); } else { echo (" $y is greater than $x"); }?> Then the output on the screen: 5 is greater than 4.5 Program 2.3 PHP program to find the greatest of three numbers using if else if else statement The syntax of if else if else statement is: if (this condition is true) { print this statement; } else if (this condition is true) { print this statement; } else { print this statement; } <?php $x = 4.5; $y = 5; $z = 6; if ($x > $y & & $x > z){ echo (" $x is greater than $y and $z"); } else if ($y>$z & & $y>x){ echo (" $y is greater than $x and $z"); } else { echo (" $z is greater than $x and $y"); }?> Output on the screen: 6 is greater than 4.5 and 5 Program 2.4 PHP program to print the first ten natural numbers using for loop statement <?php for ($i=1; $i<=10; $i++) echo (" \n $i");?> Output on the screen: 1 2 3 4 5 6 7 8 9 10 for ($i=1; $i<=10; $i++) denote the for loop statement for PHP and the syntax of the for loop statement is: for (initialization; condition; increment) Here: $i=1 denote initialization (i.e., from where to start) $i<=10 denote the condition (i.e., stop when the number 10 is reached) $i++ imply increment (which tells the value of $i to increase by 1 each time the loop is executed) and $i++ is the same as $i+1. Since the initialization i.e., $i=1 The statement: echo (" \n $i"); make provision to print the output: 1 on the screen. After this, the following execution takes place: Now, $i= 1 Is the condition ($i<=10) is true? Yes because $i=1 Do this $i= 1+1 = 2 The statement echo (" \n $i"); make provision to print the output: 2 Now, $i= 2 Is the condition ($i<=10) is true? Yes because $i=2 Do this $i= 2+1 = 3 The statement echo (" \n $i"); make provision to print the output: 3 Now, $i= 3 Is the condition ($i<=10) is true? Yes because $i=3 Do this $i= 3+1 = 4 The statement echo (" \n $i"); make provision to print the output: 4 Now, $i= 4 Is the condition ($i<=10) is true? Yes because $i=4 Do this $i= 4+1 = 5 The statement echo (" \n $i"); make provision to print the output: 5 Now, $i= 5 Is the condition ($i<=10) is true? Yes because $i=5 Do this $i= 5+1 = 6 The statement echo (" \n $i"); make provision to print the output: 6 Now, $i= 6 Is the condition ($i<=10) is true? Yes because $i=6 Do this $i= 6+1 = 7 The statement echo (" \n $i"); make provision to print the output: 7 Now, $i= 7 Is the condition ($i<=10) is true? Yes because $i=7 Do this $i= 7+1 = 8 The statement echo (" \n $i"); make provision to print the output: 8 Now, $i= 8 Is the condition ($i<=10) is true? Yes because $i=8 Do this $i= 8+1 = 9 The statement echo (" \n $i"); make provision to print the output: 9 Now, $i= 9 Is the condition ($i<=10) is true? Yes because $i=9 Do this $i= 9+1 = 10 The statement echo (" \n $i"); make provision to print the output: 10 stop because the condition $i<=10 is achieved. If you replace the statement for ($i=1; $i<=10; $i++) by the statement for ($i=1; $i==10; $i++) Then there will be no display of output on the screen. If the statement for ($i=1; $i<=10; $i++) is replaced by the statement for ($i=1; $i=10; $i++) Then the output on the screen is: 10 10 10 10 10 10 10 10 10 10 10 10 10 continues Program 2.5 PHP program to print the first ten natural numbers using while loop statement The syntax of while loop statement is: while (this is the condition) { execute this statement; } <?php $i = 1; while ($i <= 10) { echo "\n $i "; $i++; }?> Output on the screen: 1 2 3 4 5 6 7 8 9 10 Program 2.6 PHP program to print the first nine natural numbers using do while loop statement The syntax of do while loop statement is: do { execute this statement; } while (this is the condition); <?php $i = 1; do { echo "\n $i "; $i++; } while ($i <= 9);?> Output on the screen: 1 2 3 4 5 6 7 8 9 Program 2.7 PHP program to print the average of the first10 numbers using for loop statement <?php $i; $avg; $sum = 0; for ($i=1; $i<=10; $i++) $sum = $sum + $i; $avg = $sum/10; echo "\n sum of the first 10 numbers = $sum "; echo"\n average of the first 10 numbers = $avg ";?>

Output on the screen: sum of the first 10 numbers = 55 average of the first 10 numbers = 5.5 Program 2.8 Switch case method a) <?php $ch ='3'; switch ($ch) { case '1': echo "Red"; break; case '2': echo "White"; break; case '3': echo "Yellow"; break; case '4': echo "Green"; break; default: echo "Error"; break; }?> Output on the screen: Yellow b) <?php $ch ='birds'; switch ($ch) { case 'animal': echo "elephant"; break; case 'reptiles': echo "crocodile"; break; case 'birds': echo "parrot"; break; case 'mammals': echo "cow"; break; default: echo "Error"; break; }?> Output on the screen: parrot Program 2.9 Addition of two numbers using PHP function <?php function addition ($a, $b) { return $a + $b; } $sum = addition (4, 3); echo "the sum of two numbers = $sum ";?> Output on the screen: the sum of two numbers = 7 JavaScript JavaScript / Jscript (designed by an American technologist andco-founder of the Mozilla project, the Mozilla Foundation, and the Mozilla Corporation "Brendan Eich" and it was developed under the name Mocha, the language was officially called LiveScript when it first shipped in beta releases of Netscape Navigator 2.0 in September 1995, but it was renamed JavaScript) is an relatively popular object-oriented scripting interpreted programming language embedded in high level programming language of Hypertext Markup Language (the language that websites are rendered in and basically, everything you and your readers see on the "front-end" is HTML), commonly abbreviated as HTML pages primarily used to design interactive websites with dynamic content and perform functions that the HTML cannot do, because of its reliability, simplicity and easy to understand, easy to use, write, modify and debug and quick to learn. A Simple JavaScript program to print the word "Hello World!" on screen: <!DOCTYPE html> <html> <body> <script> document.write ("Hello World!"); </script> </body> </html> In the above example: <!DOCTYPE html> <html> <body> </body> </html> denote: HTML document and <script> document.write ("Hello World!"); </script> denote JavaScript code to print out the string Hello World! on the screen. In the above example, document.write () denote function or method which print out the string Hello World! on the console screen. If you fail to include the tag <script> </script>in the above example i.e., <!DOCTYPE html> <html> <body> document.write ("Hello World!"); </body> </html> Then document.write ("Hello World!"); will be outputted on the screen instead of Hello World!. If you replace the text within the double quotation marks by the word hello i.e., <!DOCTYPE html> <html> <body> document.write ("hello"); </body> </html> Then the output on the screen is: hello Note: Even if you write <SCRIPT> </SCRIPT> instead of <script> </script>. There will be no change in the output on the screen. But if you write Document.write ("hello"); instead of document.write ("hello"); There will be no display of the output on the screen. Even if you replace the tag: <script> by: <script type="text/javascript"> i.e., <!DOCTYPE html> <html> <body> <script type="text/javascript"> document.write ("Hello World!"); </script> </body> </html> There will be no change in the output on the screen i.e., Hello World will be displayed on the console screen. Program 1.1 a) <!DOCTYPE html> <html> <body> <script> document.write ("Hello World!"); document.write ("Hello World!"); </script> </body> </html> Output on the screen: Hello World!Hello World! b) <!DOCTYPE html> <html> <body> <script> document.write ("\n Hello World!"); document.write ("\n Hello World!"); </script> </body> </html> Output on the screen: Hello World! Hello World! c) <!DOCTYPE html> <html> <body> <script> document.write ("<br>Hello World!</br>"); document.write ("Hello World!"); </script> </body> </html> Output on the screen: Hello World! Hello World! d) <!DOCTYPE html> <html> <body> <script> document.write ("<br>Hello World!</br>"); document.write ("<br>Hello World!</br>"); </script> </body> </html> Output on the screen: Hello World! Hello World! e) <!DOCTYPE html> <html> <body> <script> document.write ("<b><br>Hello World!</br></b>"); document.write ("<br>Hello World!</br>"); </script> </body> </html> Output on the screen: Hello World! Hello World! f) <!DOCTYPE html> <html> <body> <script> document.write ("<i><b><br>Hello World!</br></b></i>"); document.write ("<br>Hello World!</br>"); </script> </body> </html> Output on the screen: Hello World! Hello World! Program 1.2 JavaScript program to add two numbers: <!DOCTYPE html> <html> <body> <p>A typical addition operation adds two numbers and produces a new number.</p> <script> var x; var y; var z; x =100; y = 200; z = x+ y; document.write (" The sum of two numbers is: " + z); </script> </body> </html> Output on the screen: A typical arithmetic operation takes two numbers and produces a new number. The sum of two numbers is: 300 The statements: var x; var y; var z; imply: that we are creating the variables x, y & z. Equal sign implies: storage operator The statements: x=100; y = 200; z = x+ y; imply: that we are storing the values to the created variables (i.e., we are storing the value 100 for x and 100 for y and x + y for z). The statement: document.write (" The sum of two numbers is: " + z); make provision to print the output: The sum of two numbers is: 200 on the screen. In the statement: document.write (" The sum of two numbers is: " + z); There are two strings: The sum of two numbers is: z plus operator (+) functions as the concatenation operator (concatenation means connecting two statements to produce a single statement) which (here) concatenates the string: "The sum of two numbers is: " and the string: "z (which is 100+ 100 =200) " -- producing a String statement The sum of two numbers is: 200, which is displayed on the screen as the result. The statement: <p>A typical addition operation adds two numbers and produces a new number.</p> make provision to print the output: A typical arithmetic operation takes two numbers and produces a new number. on the screen. Note: If the statement: <p>A typical addition operation adds two numbers and produces a new number.</p> is replaced by the statement: <h1>A typical addition operation adds two numbers and produces a new number.</h1> Then the output on the screen is: A typical arithmetic operation takes two numbers and produces a new number. The sum of two numbers is: 200 Program 1.3 JavaScript program to subtract two numbers: <!DOCTYPE html> <html> <body> <h1>A typical subtraction operation subtracts two numbers and produces a new number.</h1> <script> var x; var y; var z; x=300; y = 200; z = x- y; document.write (" The difference of two numbers is: " + z); </script> </body> </html> Output on the screen: A typical subtraction operation subtracts two numbers and produces a new number. The difference of two numbers is: 100 Program 1.4 JavaScript program to divide two numbers: <!DOCTYPE html> <html> <body> <p>A typical division operation divides two numbers and produces a new number.</p> <script> var x; var y; var z; x=300; y = 200; z = x/ y; document.write (" The division of two numbers is: " + z); </script> </body> </html> Output on the screen: A typical division operation divides two numbers and produces a new number. The division of two numbers is: 1.5 Program 1.5 JavaScript program to multiply two numbers: <!DOCTYPE html> <html> <body> <p>A typical multiplication operation multiplies two numbers and produces a new number.</p> <script> var x; var y; var z; x=300; y = 200; z = x* y; document.write (" The multiplication of two numbers is: " + z); </script> </body> </html> Output on the screen: A typical multiplication operation multiplies two numbers and produces a new number. The multiplication of two numbers is: 60000 Program 1.6 JavaScript program to find the area of a circle <!DOCTYPE html> <html> <body> <script> var r; var area; r=3; area = 4*3.14* r* r; document.write (" The area of the circle is: " + area +"\n centimeter square"); </script> </body> </html> Output on the screen: The area of the circle is: 113.03999999999999 centimeter square Program 1.7 JavaScript program to find the square root of a number <!DOCTYPE html> <html> <body> <script> var x; var z; x=4; z = Math.sqrt (x); document.write (" The square root of a number z is: " + z); </script> </body> </html> Output on the screen: The square root of a number z is: 2 Program 1.8 JavaScript program to find the square of a number <!DOCTYPE html> <html> <body> <script> var x; var z; x=4; z = x*x; document.write (" The square of a number z is: " + z); </script> </body> </html> Output on the screen: The square of a number z is: 16 If the statement: z = x*x; is replaced by: z = Math.pow ((x), 2); i.e., if the above program is rewritten as: <!DOCTYPE html> <html> <body> <script> var x; var z; x=4; z = Math.pow ((x), 2); document.write (" The square of a number z is: " + z); </script> </body> </html> Then there will be no change in the output on the screen i.e., The square of a number z is: 16 will be outputted on the screen. Which means: z = pow ((x), 2); is the same as z = x*x; Program 1.9 JavaScript program to find the cube root of a number <!DOCTYPE html> <html> <body> <script> var x; var z; x=4; z = Math.cbrt (x); document.write (" The cube root of a number z is: " + z); </script> </body> </html> Output on the screen: The cube root of a number z is: 1.5874010519681996 Program 2.0 JavaScript program to round off a number <!DOCTYPE html> <html> <body> <script> var x; var z; x=4.5; z = Math.round (x); document.write (" The round off a number z is: " + z); </script> </body> </html> Output on the screen: The round off a number z is: 5 Program 2.1 JavaScript program to find the incremented and decremented values of two numbers. <!DOCTYPE html> <html> <body> <script> var x; var y; var z; var p; var a; var b; x=4; y=6; z=x+1; p=x-1; a = y+1; b= y-1; document.write (" The incremented value of x is: " + z); document.write (" The decremented value of x is: " + p); document.write (" The incremented value of y is: " + a); document.write (" The decremented value of y is: " + b); </script> </body> </html> Output on the screen: The incremented value of x is: 5 The decremented value of x is: 3 The incremented value of y is: 7 The decremented value of y is: 5 Program 2.2 JavaScript program to find the greatest of two numbers using if else statement The syntax of if else statement is: if (this condition is true) { print this statement using document.write function; } else { print this statement using document.write function; } <!DOCTYPE html> <html> <body> <script> var x; var y; x=4; y=6; if (x>y){ document.write (" x is greater than y"); } else { document.write (" y is greater than x"); } </script> </body> </html> Output on the screen: y is greater than x Program 2.3 JavaScript program to find the greatest of three numbers using if else if else statement The syntax of if else if else statement is: if (this condition is true) { print this statement using document.write function; } else if (this condition is true) { print this statement using document.write function; } else { print this statement using document.write function; } <!DOCTYPE html>

<html> <body> <script> var x; var y; var z; x=4; y=6; z=12; if (x>y & & x>z){ document.write (" x is greater than y and z"); } else if (y>x & & y>z) { document.write (" y is greater than x and z"); } else { document.write (" z is greater than x and y"); } </script> </body> </html> Output on the screen: z is greater than x and y Program 2.4 JavaScript program to print the first ten natural numbers using for loop statement <!DOCTYPE html> <html> <body> <script> var i; for (i=1; i<=10; i++) document.write ("" + i); </script> </body> </html> Output on the screen: 1 2 3 4 5 6 7 8 9 10 for (i=1; i<=10; i++) denote the for loop statement and the syntax of the for loop statement is: for (initialization; condition; increment) Here: i=1 denote initialization (i.e., from where to start) i<=10 denote the condition (i.e., stop when 10 is reached) i++ imply increment (which tells the value of i to increase by 1 each time the loop is executed) and i++ is the same as i+1. Since the initialization i.e., i=1 The statement document.write ("" + i); make provision to print the output: 1 on the screen. After this, the following execution takes place: i= 1 Is the condition (i<=10) is true? Yes because i=1 Do this i= 1+1 = 2 The statement document.write ("" + i); make provision to print the output: 2 Now, i= 2 Is the condition (i<=10) is true? Yes because i=2 Do this i= 2+1 = 3 The statement document.write ("" + i); make provision to print the output: 3 Now, i= 3 Is the condition (i<=10) is true? Yes because i=3 Do this i= 3+1 = 4 The statement document.write ("" + i); make provision to print the output: 4 Now, i= 4 Is the condition (i<=10) is true? Yes because i=4 Do this i= 4+1 = 5 The statement document.write ("" + i); make provision to print the output: 5 Now, i= 5 Is the condition (i<=10) is true? Yes because i=5 Do this i= 5+1 = 6 The statement document.write ("" + i); make provision to print the output: 6 Now, i= 6 Is the condition (i<=10) is true? Yes because i=6 Do this i= 6+1 = 7 The statement document.write ("" + i); make provision to print the output: 7 Now, i= 7 Is the condition (i<=10) is true? Yes because i=7 Do this i= 7+1 = 8 The statement document.write ("" + i); make provision to print the output: 8 Now, i= 8 Is the condition (i<=10) is true? Yes because i=8 Do this i= 8+1 = 9 The statement document.write ("" + i); make provision to print the output: 9 Now, i= 9 Is the condition (i<=10) is true? Yes because i=9 Do this i= 9+1 = 10 The statement document.write ("" + i); make provision to print the output: 10 stop because the condition i<=10 is achieved. If you replace the statement: for (i=1; i<=10; i++) by the statement: for (i=1; i==10; i++) Then there will be no display of output on the screen. If the statement: document.write ("" + i); is replaced by the statement document.write (<br></br> + i); Then the output on the screen is: 1 2 3 4 5 6 7 8 9 10 What will be the output of the following program: <!DOCTYPE html> <html> <body> <script> var i; for (i=1; i<=10; i++) document.write ("<br> javascript </br>"); </script> </body> </html> Answer: javascript javascript javascript javascript javascript javascript javascript javascript javascript javascript Program 2.5 JavaScript program to print the first ten natural numbers using while loop statement The syntax of while loop statement is: while (this is the condition) { execute this statement; } <!DOCTYPE html> <html> <body> <script> var i=1; while (i<=10) document.write ("" + i++); </script> </body> </html> Output on the screen: 1 2 3 4 5 6 7 8 9 10 Program 2.6 JavaScript program to print the first nine natural numbers using do while loop statement The syntax of do while loop statement is: do { execute this statement; } while (this is the condition); <!DOCTYPE html> <html> <body> <script> var i=1; do{ document.write ("" + i++); } while (i<10) </script> </body> </html> Output on the screen: 1 2 3 4 5 6 7 8 9 Program 2.7 JavaScript program to print the average of the first10 numbers using for loop statement <!DOCTYPE html> <html> <body> <script> var i, avg, sum = 0; for (i=1; i<=10; i++) sum = sum + i; avg = sum/10; document.write ("<br> sum of the first 10 numbers = </br>" + sum); document.write ("<br> average of the first10 numbers = </br>" + avg); </script> </body> </html> Output on the screen: sum of the first 10 numbers = 55 average of the first10 numbers = 5.5 Program 2.8 Switch case method a) <!DOCTYPE html> <html> <body> <script> var ch ='2'; switch (ch) { case '1': document.write ("Red"); break; case '2': document.write ("White"); break; case '3': document.write ("Yellow"); break; case '4': document.write ("Green"); break; default: document.write ("Error"); break; } </script> </body> </html> Output on the screen: White b) <!DOCTYPE html> <html> <body> <script> var ch ='animal'; switch (ch) { case 'animal': document.write ("elephant"); break; case 'reptiles': document.write ("crocodile"); break; case 'birds': document.write ("parrot"); break; case 'mammals': document.write ("cow"); break; default: document.write ("Error"); break; } </script> </body> </html> Output on the screen: elephant Program 2.9 Addition of two numbers using JavaScript function <!DOCTYPE html> <html> <body> <script> function addition (a, b) { return a + b; } document.write ("" + addition (4, 3)); </script> </body> </html> Output on the screen: What is the mistake in the following program: <!DOCTYPE html> <html> <body> <script> function addition (a, b) { return a % b; } document.write ("" + function (4, 3)); </script> </body> </html> Answer: There is mistake in the above program. return a % b; is written instead of: return a + b; The program: <html> <head> <script> function addNumbers () { var x = parseInt

(document.getElementById ("value1").value); var y = parseInt (document.getElementById ("value2").value); var z = document.getElementById ("answer"); z.value = x + y; } </script> </head> <body> value1 = <input type ="text" id="value1"/> value2 = <input type ="text" id="value2"/> <input type="button" value="Click here" onclick="addNumbers ()"/> Answer = <input type="text" id = "answer"/> </body> </html> Corresponds to the output: The above program demonstrates how to get two inputs from the user (i.e., value1 and value2) and have a button (i.e., click here) on the page to call a function: function addNumbers () to process the inputs and output the answer on the web screen. var x = parseInt (document.getElementById ("value1").value); denote: that we are creating the variable x and assigning it a value = parseInt (document.getElementById ("value1").value) parseInt (document.getElementById ("value1").value) get the value (i.e., get the integer value) for x entered in the input field which is defined with id value1. parseInt (document.getElementById ("value2").value) get the value (i.e., get the integer value) for y entered in the input field which is defined with id value2. After entering the values for x and y in the input fields you click on the button click here after button click "on click" in the statement: <input type="button" value="Click here" onclick="addNumbers ()"/> call the function addNumbers () within which var z = document.getElementById ("answer"); z.value = x + y; is executed i.e., the entered values for x and y are added and the answer is entered in the input field which is defined with id answer. Note: if you want to enter the floating values (say if you want enter 1.63 & 1.569) for x and y in the input fields, then you need to replace the statements: var x = parseInt (document.getElementById ("value1").value); var y = parseInt (document.getElementById ("value2").value); in the above program by the statements: var x = parseFloat (document.getElementById ("value1").value); var y = parseFloat (document.getElementById ("value2").value); i.e., <html> <head> <script> function addNumbers () { var x = parseFloat (document.getElementById ("value1").value); var y = parseFloat (document.getElementById ("value2").value); var z = document.getElementById ("answer"); z.value = x + y; } </script> </head> <body> value1 = <input type="text" id="value1"/> value2 = <input type="text" id="value2"/> <input type="button" value="Click here" onclick="addNumbers ()"/> Answer = <input type="text" id = "answer"/> </body> </html> What will be the output of the following programs: a) <html> <head> <script> function area () { var r = parseFloat (document.getElementById ("value1").value); var a = document.getElementById ("answer"); a.value = 4*3.14* r* r; } </script> </head> <body> radius = <input type="text" id="value1"/> <input type="button" value="Click here" onclick="area ()"/> Area of the circle = <input type="text" id = "answer"/> </body> </html> Answer: b) <html> <head> <script> function ifelse () { var x = parseInt (document.getElementById ("value1").value); var y = parseInt (document.getElementById ("value2").value); var z = parseInt (document.getElementById ("value3").value); if (x>y & & x>z) { document.write (" x is greater than y and z"); } else if (y>x & & y>z) { document.write (" y is greater than x and z"); } else { document.write (" z is greater than x and y"); } } </script> </head> <body> x = <input type="text" id="value1"/> y = <input type="text" id="value2"/> z = <input type="text" id="value3"/> <input type="button" value="Click here" onclick="ifelse ()"/> </body> </html> Answer: c) <html> <head> <script> function switchcase () { var ch = parseInt (document.getElementById ("value1").value); switch (ch) { case 1: document.write ("Red"); break; case 2: document.write ("White"); break; case 3: document.write ("Yellow"); break; case 4: document.write ("Green"); break; default: document.write ("Error"); break; } } </script> </head> <body> ch = <input type="text" id="value1"/> <input type="button" value="Click here" onclick="switchcase ()"/> </body> </html> Answer: The difference between Java and JavaScript Both Can Run in a Browser. But JavaScript program is often faster, sometimes almost instant. Java programs take a little bit of time (several seconds or more) to process. Java programs consumes a lot of computer memory to function properly, which can cause a computer to slow down or another program to operate more slowly while JavaScript programs uses less memory (very little in some cases) to do its processing and function properly. JavaScript contains a much smaller and simpler set of commands than does Java. It is easier and faster for the average beginner to understand and learn. Java codes is typically written in an Integrated Development Environment (IDE) and compiled into class files and then to byte codes and then executed by JVM (Java Virtual Machine) -- while JavaScript code is directly executed by browser. One more difference which comes from this fact is that, Java is run inside Java Virtual Machine and needs Java Development Kit or Java Runtime Environment for running, on the other hand JavaScript runs inside browser and almost every modern browser (such as Chrome, Mozilla) supports JavaScript. Operating System: A well-defined set of instructions in the form of statements that is installed into the computer which provide instructions for computer how to operate (i.e., how to receive the raw data through input devices (like key board, mouse etc.), process the input data through processing device called CPU (Central Processing Unit) and store the processed data (in information storage devices like hard disks) and display the processed data through output devices (like monitor, printer etc.)). A well-defined instruction is called a code and a well-defined set of instructions constitute a program (i.e., compilation of codes gives a program). For example: word is a code and a paragraph is a program (i.e., compilation of words gives a paragraph). Examples of Operating system (a well-defined set of instructions that is installed into the computer which provide instructions for computer how to operate) are: DOS (Disk operating system developed by Bill Gates and Paul Allen in 1980 for IBM PCs), Linux (operating system developed by Linus Torvalds at the University of Helsinki with the assistance of developers around the world), Windows NT, 95 & 2000 (developed by Microsoft corporation for PC), UNIX (developed by AT & T Bell Laboratories, Murray Hill, New Jersey) etc. Drivers: A well-defined set of instructions (what we call programs or software) that is installed into computer and stored in the form of files in the computer that allows the computer to communicate with its hardware components (i.e., hardware components like mouse, key board, printer etc.). Without drivers, the computer cannot communicate with its hardware components as a result a mouse, keyboard, or a printer wont work properly. Domain name: If we type www.google.com (which is called the domain name) in the browser, then the domain name is sent to DNS (domain name system) where the domain namewww.google.com is converted to IP address 74.125.224.72 (because website / web pages are only identified by their IP address in the server) and this IP Address is sent to the web server (a system that acts like a data center from where the required information (i.e., web page of google.com) is taken and sent to the browser and the www.google.com web page is displayed in the web browser). If you type the IP address in the browser, then DNS is not required. For human convenience (difficult remember numbers, for example: www.google.com is domain name, IP address is 74.125.224.72. Because it is difficult to remember 74.125.224.72 so www.google.com is preferred). Hosting: Host is a system that contains information and this information can be accessed by computer users by a means of internet. This process is called hosting. IP address or Internet Protocol address: Just like every house on a street has a postal address which helps the post man to find that house on a street, every computer connected to internet has an Internet Protocol address or IP address which helps the other computers to find that computer on the network. Suppose A B, C, D, E, F and G are the computers connected to each other by means of internet (i.e., they are on the network). If computer A has not assigned any IP address, then users at computers B, C, D, E, F and G cannot send any email or other data to user at computer A or user at computer A cannot receive any email or other data from the users at computers B, C, D, E, F and G by a means of internet. IP address is of four types: Public IP address and Private IP address Static and dynamic IP address Static IP address permanent IP address Dynamic IP address temporary IP address (exist only for a limited time i.e., IP address leased for a limited time). Public and Private IP address: Amazon organization is assigned an IP address IPA and Google organization is assigned an IP address IPG. And the systems (1, 2, 3, 4, 5..etc.) within the Amazon or the Google organization are assigned an IP addresses IP1, IP2, IP3 etc. IPA and IPG imply public IP addresses IP1, IP2, IP3 etc. implies private IP addresses Which means: Public IP address is used for external communication (i.e., used for the communication between the Amazon and the Google organization) and Private IP address is used for internal communication (i.e., used for communication between the systems within the Amazon or the Google organization). ASP.NET: ASP Active Server Page ASP.NET (Active Server Page Network Enabled Technology) is a technology developed by Microsoft corporation using the languages -- C#, Visual Basic. Net, J script & J# -- to build dynamic web pages / websites and web applications. Dynamic web page contains information (say date, month or year or time zone of the day) change automatically daily without a developer editing its source codes while static web page contains information (say date, month or year or time zone of the day) cannot change automatically daily without a developer editing its source codes. Virtual Memory: If the RAM (i.e., Random Access Memory) is full and it is running out of space available for storage of further information and there is no access to store further information, the idea of extending memory by using disk is called virtual memory (i.e., the further information is stored in disk and retrieved when required). This process is called paging or swapping. Server: Examples of server are: IIS (Internet Information server LATTER NAMED Internet Information service) a web server developed by Microsoft corporation, Apache HTTP (HTTP mean Hyper Text Transfer Protocol) a web server developed by Robert McCool at the national center for supercomputing applications (university of Illinois, Urbana-Champaign) to provide web hosting service. Note: XML Extensible (extendable) Markup (symbols and notations like <, >, / etc.) Language (which is both human and machine understandable language) is a simple and very flexible text format designed to store and transport data through internet. HTML (Hyper

Text Markup Language) = A text format designed to display data XML to display the output: note to people from steve jobs message Design is not just what it looks like and feels like. Design is how it works. Answer: <note> <to> people </to> <from> steve jobs </from> <message> Design is not just what it looks like and feels like. Design is how it works. </message> </note> Note: If the statement: <message> Design is not just what it looks like and feels like. Design is how it works. </message> is replaced by the statement: <Message> Design is not just what it looks like and feels like. Design is how it works. </message> Then there will be no display of the output on the console screen. The statement: <to> people </to>imply: element <to>imply: start tag and </to>imply: end tag <note> </note>is termed: parent element and <to> people </to> <from> steve jobs </from> <message> Design is not just what it looks like and feels like. Design is how it works. </message> are termed: child elements XML to display the output: Book Name of the book: Harry Potter Author: J K. Rowling Price: $255 Pages: 296 Year: 2002 Edition: 8 Answer: <Book> <Name>:Harry Potter </Name> <Author>: J K. Rowling </Author> <Price>: $255 </Price> <Pages>: 296 </Pages> <Year>: 2002</Year> <Edition>: 8 </Edition> </Book> Note: What will be the output of the following: <Book> <Name>: Harry Potter </Name> <Author> J K. Rowling </Author> <Price> $255 </Price> <rowling><Pages> 296 </Pages></rowling> <Year> 2002</Year> <Edition> 8 </Edition> </Book> Answer: Book Name of the book: Harry Potter Author: J K. Rowling Price: 255$ Pages: 296 Year: 2002 Edition: 8 Note: <rowling><Pages> 296 </Pages></rowling> is termed: child element <Pages> 296 </Pages> is termed: sub child element.

1/25/2018