DATE : 20/09/21.

C programming

How to design a C-Programme :-

Step 1 :- To open editor → gedit filename.c

ex - gedit Al_Garima_1.c

Structure of C-Programme.

1. Header File Inclusion.

collection of predefined functions.
or
Readymade function or Library function or Built in function

Ex.
- stdio.h (Standard input output)
- math.h
- conio.h (console input output)

Syntax.

preprocessor

#include < Header file name > → Closing angular bracket

↓ directive

↓ opening angular bracket

or

#include " Header file name "

↓ double quotes

Step 2: int main ( ) → pair of parenthesis

{ → curly braces

—— BODY

——

return 0 ;

} → curly braces

To print message

printf( ) → print format

Syntax:

printf ("Your message") ;

Step 3: To compile → gcc filename.c or gcc filename.c -O target Name.

↓

GNU Compiles Collection

↓

GNU Not Unix.

Que- Write a program to print your name.

Input:

```
#include<stdio.h>
int main ()
{
printf ("Oraima Mishra")
}
```

Output:

Oraima Mishra.

Escape Sequence:

1) \n → next line or new line
2) \t → Horizontal Tab
3) \\ → To print \
4) \b → backspace.
5) \r → carriage return — compiles dependent
6) \v → vertical Tab — only on gcc compiler
7) \a → audible signal
8) \" → To print "

C Programming.

Developed by Denis Ritchie at AT&T Bell lab USA in 1972

History

1960 ← ALGOL → Algorithm Logic → Europe

↓

Typless ← 1967 ← BCPL → Basic combined Programming language.

+

Direct Memory Access

↓

B Programming → 1970

↓

1970 ← C Programming → Denis Ritchie

Need of C Programming

To design UNIX Operating System.

Uses of C :-

* Mostly used in designing or developing system software like operating system, utilities, Drivers (sound drivers), libraries, Kernal, etc.
* For programming embedded system, IOT devices and gaming.
* Less frequently used for Application Programming.
* Serves as a intermediate language for new languages like C++, Java, C#, etc.

Features or characteristics of C :

* High level language and hence easy to understand, read, write, debug, etc.
* Small size language → only 32 Keywords.
* Case sensitive language like Eat and eat both are different.
* Quick / fast language because of use of pointers.
* Portable but platform dependent.
* Functional / structured / Procedural language.
* Extensible language.
* Tightly coupled with datatype.

How to learn C :

1. • Alphabet (A-Z), (a-z)
   • Digit (0-9)
   • Special Symbol like *, -, #, etc.
   → 2. Keywords. Identifiers. Variables. Constants. (128 Total) → 3. Instructions → 4. Program

→ Keywords :

* These are the reserved words.
* They convey specific meaning.
* There are 32 words in total.
* Ex:- auto, break, case, char, continue, do, default, const, double, else, enum, extern, for, if, goto, float, int, long, register, return, signed, static, sizeof, short, struct, switch, typedef, union, void, while, volatile, unsigned.

→ Identifier :

A name given to any object or entity for identification or recognition purpose.

Rules for naming Identifier :-

* Alphabet + digit + Underscore (_) only as a special symbol
  A-Z, a-z 0-9
  Total 63

* NO Keywords is allowed.
* First character of identifier cannot be a digit.
* Spaces are not allowed.
* We can have maximum upto 32 character length

Example :- (1) Int → ✓
(2) float → ✓
(3) struct → ✗
(4) hello_hii$ → ✗

(5) hello hii → X
(6) _hello/hii → ✓
(7) 1-hello → X
(8) __ab2 → ✓
(9)
(10)

→ Variable :
Named Memory location that can store some value and that value can vary or change time to time.

Syntax.
Datatype variablename ;

Ex. → int a ;

→ Constant :
Named memory location that stores a fixed value
Declaration and Initialisation

Syntax.
const datatype constantname = value ;

Ex. → const float pi = 3.14 ;

pi
3.14

Note : 1 Bit → 0 (OFF) / 1 (ON)
↓
4 bits = Nibble
↓
8bits = 1 Byte
↓
1024 Byte = 1 Kilobyte
↓
1024 KB = 1 Megabyte
↓
1024 MB = 1 Gigabyte
↓
1024 GB = 1 Terabyte.

Note : ASCII → American Standard Code for Information Interchange.

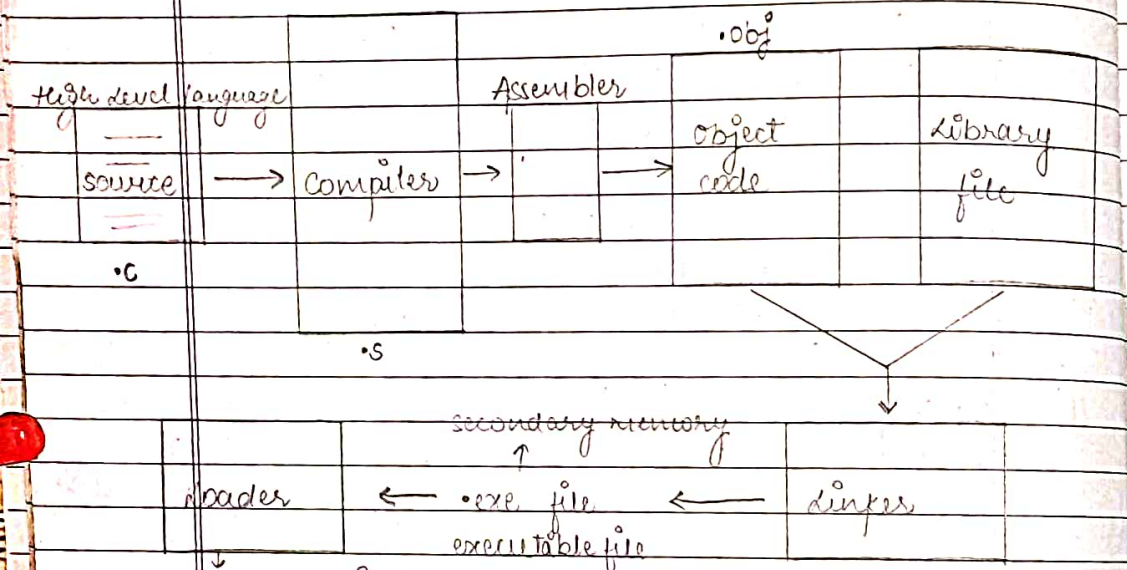
ASCII Table

0 - 47 → Special Characters
48 - 57 → Digits 0-9
58 - 64 → S.C.
65 - 90 → Alphabet Upper Case A-Z
91 - 96 → S.C.
97 - 122 → Alphabet Lower Case a-z
123 - 127 → S.C.

1 Byte = 8 bits = ⊖ — — — — — — — ⊖
MSB ↓ Sign bit → +ve = 0 / -ve = 1
LSB ↓

Datatype in C :-

S.no.	Datatype	Keyword	Size	Range
1.	Valueless / No type	void	— NA —	— NA —
2.	Integer Character	char / signed char	1B	-2^7 to 2^7-1 -128 to 127
		unsigned char	1B	0 to 255
3.	Integer	short int / short / signed short int / signed short	2B	-32768 to +32767
		unsigned short / unsigned short int	2B	0 to $2^{16}-1$ 0 to 65535
		int / signed int	2B (16 or 32) 4B (64)	-32768 to +32768 -2^{31} to $+2^{31}-1$
		unsigned int	2B (16 or 32) 4B (64)	0 to 65535 0 to $2^{32}-1$
		long int / long / signed long int / signed long	4B (16 or 32) 8B (64)	-2^{31} to $+2^{31}-1$ -2^{63} to $+2^{63}-1$
		unsigned long int / unsigned long	4B (16 or 32) 8B (64)	0 to $2^{32}-1$ 0 to $2^{64}-1$
		long long int / long long / signed long long int / signed long long	8B (16 or 32) 12B (64)	-2^{63} to $+2^{63}-1$ -2^{95} to $+2^{95}-1$
		unsigned long long int / unsigned long long	8B (16 or 32) 12B (64)	0 to $2^{64}-1$ 0 to $2^{96}-1$
4.	Floating point	float	4B	$-3.4E+38$ to $-3.4E-38$ or $-3.4E-38$ to $-3.4E+38$
		double	8B	$-1.7E+308$ to $-1.7E-308$ or $+1.7E-308$ to $+1.7E+308$
		long double	12B	$-3.4E+4932$ to $-3.4E-4932$ or $+3.4E-4932$ to $+3.4E+4932$



Compiler
Programme which will convert High level language to low level language in a single go.

Assembler
Programme which will convert assembly language to ELL.

Linker
Programme which will link object code with predefine library files and will generate executable file.

Loader
It is a part of operating system which is responsible for loading executable file from secondary storage to primary storage for execution purpose.

→ **Format Specifier :-**

char	%c
short int	%hd or %hi
unsigned short int	%hu
int	%d or %i
unsigned int	%u
long int	%ld or %li
unsigned long int	%lu
long long int	%lld or %lli
unsigned long long int	%llu
float	%f
double	%lf
long double	%Lf
octal	%o (not zero)
hexadecimal	%x or %X
string	%s
exponential	%e or %E

Collection of characters

Note :
- %f → After decimal upto 6 places.
- %lf → " 14 " .
- %lf → " 19 " .
- For upto 2 decimal place → %.2f.

Variable

Datatype variablename ;

Speed
PAGE NO.:
DATE : / /

Syntax

printf("Your message with escape sequence and format specifier", Name of variable);

Que- WAP to ~~input~~ add two nos. 12 and 5.

1) Input:

```
#include<stdio.h>
int main ( )
{
  int a=12, b=5, s ;
   s=a+b ;
  printf(" %d + %d = %d", a,b,s );
}
```

Output

12 + 5 = 17.

2) Input

```
# include<stdio.h>
int main ( )
{
  int a=12, b=5, s ;
   s= a+b ;
  printf(" %d", s);
}
```

Output.

17.

3) Input

```
#include <stdio.h>
int main ( )
{
  int a=12, b=5, s ;
  s = a+b ;
printf(" sum of %d + sum of %d = \n %d", a,b,s );
}
```

Output

sum of 12 + sum of 5 =
17.

→ Scan format

scanf ( ) ;

Syntax.

scanf("format specifiers", address of variable);
↓
& (Empercent operator)

Ex. Input

```
# include<stdio.h>
int main ()
{
  int a,b,s ;
scanf ("%d %d", &a, &b);
   s = a+b ;
printf ("Sum of %d and %d = \n %d", a,b,s );
}
```

Output

Sum of a and b =
5

Que - WAP to input Principal, Rate & Time and calculate simple interest.

Input

```
#include <stdio.h>
int main ()
{
    float p,r,t,s ;
    printf("Enter Principal, Rate and Time respectively");
    scanf(" %f %f %f ", &p, &r, &t );

        s = p*r*t / 100 ;

    printf (" %f",s );
}
```

Output.

Enter Principal, Rate and Time

Que- WAP to input the radius of circle and calculate Area and perimeter.

Input

```
#include <stdio.h>
  int main ()
{
    const float pi = 3.14 ;
    float r, A, P ;
    printf(" Enter Radius");
    scanf(" %f", &r);

    P = 2*pi*r ,  A = pie*r*r ;

    printf (" %f %f", P, A);
}
```

Note :- pow (A, B) = A^B

ex - pow (2,5) = 2^5

also use #include <math.h>

Que- WAP to input the sides of a triangle and calculate area using Heron's formula.

```
#include <math.h>
#include <stdio.h>
  int main ( )
{
    float a,b,c,s, A ;
    printf(" Enter the sides of the triangle ");
    scanf (" %f %f %f ", &a, &b, &c);

    S = (a+b+c) / 2     , A = pow(s(s-a)(s-b)(s-c), 0.5) ;
```

```
    printf("%f", A);
}
```

Que. WAP to input P,R,T & calculate compound interest.

```
#include<math.h>
#include <stdio.h>
 int main ()
{
   float c,p,r,t;
   printf("Enter Principle, Rate and Time respectively");
   scanf("%f %f %f", &p, &r, &t);

      c = ptpow((1+r/100), t) - p ;

   printf("%f", c);
}
```

Q. Input

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

Output

Hello5

Que. Input:

```
#include <stdio.h>
  int main ()
{
    int x ;
    x = printf("Hello\n Hii");
    printf("%d", x);
}
```

Output

Hello
Hii 8

Note:- * printf print no. of characters.
* scanf print no. of inputs.

Que. Input

```
#include <stdio.h>
int main ()
{
  int x, y=1205;
  x = printf("%d", y);
  printf("%d", x);
}
```

Output

12054

Que. Input

```
#include <stdio.h>
int main ()
{
  int a, b, c;
  c= scanf("%d %d", &a, &b);
  printf("%d", c);
}
```

Output

2

Que. Input

```
#include <stdio.h>
int main ()
{
  char c=123;
  printf("%d", c);
}
```


Output

123.

Que. Input

```
#include<stdio.h>
int main ()
{
  char c=128;
  printf("%d", c);
}
```

Output

-128

Que. Input

```
#include <stdio.h>
int main ()
{
  char c=129;
  printf("%d", c);
}
```

Output

-127

Que. Input

```
#include <stdio.h>
int main ()
{
  short int x=32770;
```

```
printf("%hd", X);
}
```

Output:
-32766.

Lab.

Que. Suppose you are having a piggy bank with following currencies & their corresponding no. of coins

Currencies	Rs 1/-	Rs 2/-	Rs 5/-	Rs 10/-
No. of Coins	7	10	15	12

Find the total amount in the piggy bank.

Input
```
#include <stdio.h>
int main ()
{
int a=1, b=2, c=5, d=10, s;
s= a*1 + b*10 + c*15 + d*10;
printf("%d", s);
}
```

Output
222.

Que- Suppose an employee have basic salary Rs- 39576/-
HRA is 10% of basic salary, DA is 7% of basic salary,
TA is 12% of B.S. Calculate total salary.

Input
```
#include <stdio.h>
int main ()
{
float bs=39576, hr, da, ta, s;
hr= 0.1*bs, da= 0.07*bs, ta= 0.12*bs;
scanf("%f %f %f %f", &bs, &hr, &da, &ta);
s= bs+hr+ta+da;
printf("%f", s);
}
```

Output

Que WAP to input your mobile no. & display it.

Input
```
#include <stdio.h>
int main ()
{
double a;
printf("Enter Mobile Number");
scanf("%lf", &a);
printf("%lf", a);
}
```

Speed PAGE NO.: DATE: / /

Que- WAP to input a character & print its ASCII value.
A=65 , a=97 , 0=48 , $=36.

(I) A=65

Input
```
#include <stdio.h>
int main ()
{
    char s='A';
    printf("%d", s);
}
```

Output
65.

Input
```
#include<stdio.h>
int main ()
{
    char s=65;
    printf("%c", s);
}
```
output
A

(II) a=97

Input.
```
#include <stdio.h>
int main ()
{
    char s='a';
    printf("%d", s);
}
```

Output.
97.

(III) 0=48 (zero)

Input
```
#include <stdio.h>
int main ()
{
    char s='0';
    printf("%d", s);
}
```

Output-
48.

(IV) $=36

Input
```
#include<stdio.h>
int main ()
{
    char $='$';
    printf("%d", $);
}
```

Output.
36

Que- Suppose initial population of a city is 1248000. If increase by 10% during 1st yr and then inc. by 12% during 2nd yr. Find the population after 2nd yr.

Input
```
#include<stdio.h>
int main ()
{
    int a=1248000, b, c, d, e;
    b=0.1*a;
    c=a+b;
    d=0.12*c;
    e=c+d;
    printf("Total population=\n%d", e);
}
```

Output.
Total Population=
1537536.

Speed
PAGE NO.:
DATE : / /

Que- Suppose Raju got 6 & $\frac{1}{2}$ apples from each of Raghu, Shiv & Akash. Calculate total no. of apples he having without adding them.

Input

```
#include<stdio.h>
int main ()
{
    float a=3, b=6.5, c;
    c= a*b;
    printf("%f", c);
}
```

Comments :
To make the code user friendly.

(I) Single line comment (II) Multiline comment

Syntax: //comment Syntax: /* —
—
— */.

Ex.

```
#include <stdio.h>
int main ()
{
    float pi= 3.14; //Here pi is universal constant
    float A= pi*3*3; //Here A will store area of circle.
    printf ("%f", r);
}
```

Note: Compiles will not read the comments, this is only for programmer and users.

Que- Input

```
#include <stdio.h>
int main ()
{
    int x= 315;
    printf("%d %X %x %o", x, x, x, x);
}
```

Output

315 13B 13b 473.

%X → Hexadecimal

$(315)_{10} = (13B)_{16}$

10 (0-9)

16 (0-15)

16	315	
16	19	11
16	1	3
	0	1

0-9
10 → A or a
11 → B or b
12 → C or c
13 → D or d
14 → E or e
15 → F or f

$(315)_{10} = (473)_8$

8	315	
8	39	3
8	4	7
	0	4

Que. Input

```
#include<stdio.h>
int main()
{
    int X = 0315;   // octal
    printf("%d %X %x %o", X, X, X, X);
}
```

Output

205 CD cd 315

* $(315)_8 = (205)_{10}$

$= 3*8^2 + 1*8^1 + 5*8^0$

$= 192 + 8 + 5$

$= 205.$

$(205)_{10} = (CD)_{16}$

16	205	
16	12	13 → D
	0	12 → C

* $(315)_{23} = (011001101)_2$

011 001 101

2^2	2^1	2^0
4	2	1
5 → 1	0	1
1 → 0	0	1
3 → 0	1	1

$(1100\,1101)_2 = (CD)_{16}$

2^3	2^2	2^1	2^0	
8	4	2	1	
1	1	0	1	→ D
1	1	0	0	→ C

Que Input

```
#include<stdio.h>
int main()
{
    int X = 0X3A;   // hexadecimal
    printf("%d %X %x %o", X, X, X, X);
}
```

Output

58 3A 3a 72

DATE : / /

$(3A)_{16} \rightarrow (?)_{10}$

$3 \times 16^1 + A \times 16^0$

$48 + 10$

58

$(3A)_{16} = (58)_{10}$

$(58)_{10} = (?)_8$

$(58)_{10} = (72)_8$

8	58
8	7 ↑ 2 ↑
	0 → 7

Speed

PAGE NO.:

DATE : / /

Operator

Operator are symbols which provide a specific result after applying on some data values.

Operants

Operants are the data values on which operators are being applied to get a particular result.

Expression

It is the collection of operator and operants.

Ex- $C = A * B + 7$ → Expression

operator = =, *, +

Operants = A, B, 7 (could be any const. & vari.)

Category of Operators :-

(I) Arithmetic Operator (B)

(II) Relational " (B)

(III) Equality " (B)

(IV) Logical (&&, ||, !) (B) (U)

(V) Assignment & short hand assignment (B)

(VI) Bitwise operator (<<, >>, &, |, ^, ~) (B) (U)

(VII) Conditional / Ternary op. (T)

(VIII) Unary plus and unary minus (U)

(IX) Increment & Decrement op. (p++, ++p, p--, --p) (U)

(X) Comma op. (,) (B)

(XI) Sizeof (B)

DATE : / /

(XII) Miscell. op.

Types of operators :-

(I) Unary op.
(II) Binary op.
(III) Ternary op.

[1] Arithmetic Operator [Binary] (*, +, -, /, %)

S.No.	Operation	Operator	Expression	Result
1.	Addition	+ (int, char, float)	C = 5+8	C = 13
2.	Subtraction	- (int, char, float)	C = 5-8	C = -3
3.	Multiplication	* (int, char, float)	C = 5*8	C = 40
4.	Division	/ (int, char, float)	C = 13/4	C = 3 (quotient)
5.	Modulus	% → char, int ✓ ; float ✗	C = 13%4	C = 1 (remainder)

Precedence

If in an expression there are two or more operators of different type then the operator having highest priority will evaluate first then the other one.

*, /, % → High
+, - → low

Associativity

Order of evaluation of same procedure operators.
It could be either left to right or right to left depending on the category of operator.

Note :- For Arithmetic operator, it is left to right.

Ex -
```
#include<stdio.h>
int main ()
{
int X = 9*8+7/5-2%6;
printf("%d", X);
}
```
→ 73

9*8+7/5-2%6
72+1-2
73

Ex -
```
#include<stdio.h>
int main ()
{
int X = 8*3%2-10+9/5;
printf("%d", X);
}
```
→ -9

8*3%2-10+9/5
24%2-10+1
0-10+1
-9

Ex - WAP to find the value of A%B without using modulus operat

```
#include<stdio.h>
int main ()
{
int A, B, R;
scanf("%d%d", &A, &B);
R = A - B*(A/B); or R = A - A/B*B;
printf("%d", R);
}
```

Remainder = Dividend - Divisor × Quotient

Speed
PAGE NO.:
DATE: / /

Ex-

C = 13/4 = 3
C = -13/4 = -3
C = 13/-4 = -3
C = -13/-4 = 3

C = 13%4 = 1
C = -13%4 = -1
C = 13%-4 = 1
C = -13%-4 = -1

A = -13 B = 4
-13 - (-13/4) * 4
-13 + 12
-1

A = 13 B = -4
13 - (13/-4) * -4
13 - 12
1

A = -13 B = -4
-13 - (-13/-4) * -4
-13 - 3 * -4
-13 + 12
-1

Ex-

```
#include <stdio.h>
int main()
{
int X = 5.0%3;
printf("%d", X);
}
```

Error

Ex-

```
#include <stdio.h>
#include <math.h>
int main()
{
float X = fmod(5.5,3);
printf("%f", X);
}
```

fmod → under math.h

= 2.5

A - B * [(int)A/B]

typecast

Speed
PAGE NO.:
DATE: / /

True = 1 False = 0

Ex-

```
#include <stdio.h>
int main()
{
int X = 'B' * 3.6 + 10/3 % 7 - 5.2;
printf("%d", X);
}
```

= 2354

'B' * 3.6 + 10/3 % 7 - 5.2
66 * 3.6 + 10/3 % 7 - 5.2
237.6 + 3 % 7 - 5.2
237.6 + 3 - 5.2
240.6 - 5.2
235.4

[2] Relational Operator [Binary] [Left to Right]

S.No.	Operation	Operator	Expression	Result
1.	less than	<	C = 10 < 12	C = 1
2.	less than or equal to	<=	C = 10 <= 5	C = 0
3.	greater than	>	C = 13 > 17	C = 0
4.	greater than or equal to	>=	C = 17 >= 3	C = 1

* Precedency is same i.e. <, <=, >, >=.
* Precedency

*, /, %
+, -
<, <=, >, >=

H
↓
L

Ex-

```
#include <stdio.h>
int main()
{
int X = 10 < 7 >= 3 <= 1 > 9;
printf("%d", X);
}
```

0>=3<=1>9.;
0<=1>9.;
1>9
0

Ex- int x = 13%5<=3+9*7>1
2<=3+63>1
3<=66>1
1>1
0

Ex- int x = 7*(5>9*/4)
7*(5>2)
7*1
7

[3] Equality Operator [Left to Right] [Binary]

S.NO.	Operation	Operator	Expression	Result
1.	Equals to	==	C=5==5	C=1
			C=7==5	C=0
2.	Not equal to	!=	C=5!=5	C=0
			C=5!=7	C=1

Ex. int x = 5==5!=7
1!=7
°1

Ex. int x = 5*3==3+9>=1;
15==3+9>=1
15==12>=1
~~0>=1~~ 15==1
0

Ex- int x = 6+(5*3<9==0);
6+(15<9==0)
6+(0==0)
6+1
7

[4] Logical operator [Binary] [Left to Right]

S.NO.	Operation	Operator	Expression	Result
1.	Logical AND	&&	C=0&&0	C=0
			C=0&&1	C=0
			C=1&&0	C=0
			C=1&&1	C=1
			C=5&&0	C=0
			C=0&&-5	C=0
			C=5&&-7	C=1
2.	Logical OR	||	C=0||0	C=0
			C=0||1	C=1
			C=1||0	C=1
			C=1||1	C=1
			C=5||0	C=1
			C=0||-5	C=1
			C=5||-7	C=1

3.	logical NOT	!	C = !10	C = 1
			C = !11	C = 0
			C = !5	C = 0
			C = !-5	C = 0

Ex-
```
#include <stdio.h>
int main ()
{
    int X=7, Y=3, Z=5 , A ;
A = (X=X+Y*Z) && (Y=Y+X-Z);
printf("%d %d %d %d", X, Y, Z, A);
}
```

(7+3*5) && (3+22-5)
(7+15) && (25-5)
(22) && (20)
1

A=1 X=22 Y=20 Z=5

Ex-
```
#include <stdio.h>
int main ()
{
    int X=7, Y=3, Z=5 , A ;
A = !(X=X+Y*Z) && (Y=Y+X-Z);
printf("%d %d %d %d", X, Y, Z, A);
}
```

!(7+3*5) && (3+22-5)
!(22) && (3+22-5)
0 && (3+22-5)

X=22 Y=3 Z=5 A=0

Ex-
```
#include <stdio.h>
int main ()
{
    int X=7, Y=3, Z=5 , A ;
A = (X=X+Y*Z) || (Y=Y+X-Z);
printf("%d %d %d %d", X, Y, Z, A);
}
```

(7+3*5) || (3+22-5)
(22) ||

X=22 Y=3 Z=5 A=1

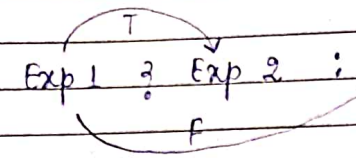
Ex-
```
#include <stdio.h>
int main ()
{
    int X=7, Y=3, Z=5, A;
A = !(X=X+Y*Z) || (Y=Y+X-Z);
printf("%d %d %d %d", X, Y, Z, A);
}
```

!(7+3*5) || (3+22-5)
!22 || (35-5)
0 || 20
1

X=22 Y=20 Z=5 A=1

[5] Conditional / Ternary operator.

Syntax :-



Que - WAP to input a no. & check no. is even or odd using c.op.

```
#include <stdio.h>
int main ()
{
    int n;
    printf(" Enter a number");
    scanf ("%d", &n);
    N%2 == 0 ? printf("Even") : printf("odd") ;
}
```

Note :- N%2 = 0 ? printf("even") : printf("Odd");
↳ error [2 value required]

Que -

```
#include <stdio.h>
int main ()
{
    int R,N;
    Scanf("%d",&N);
    R= N%2;
        R=1 ? printf("Even") : printf("Odd");
```

Always Odd.

Que - WAP to input two no.s & print the largest no.

```
#include <stdio.h>
int main ()
{
    int M,N;
    scanf("%d", &N, &M);
        N>M ? printf("%d", N) : printf("%d", M) ;
}
```

OR

```
#include <stdio.h>
int main ()
{
    int N,M;
    scanf("%d", &N, &M);
    N>M ? printf("%d", N) : N==M ? printf(" Both are
    equal & value = %d", M) : printf("%d", M);
}
```

Que - WAP to check if a person is eligible for voting or not if he/she is not eligible then also calculate after how many year that person will be eligible for voting.

```
#include <stdio.h>
int main ()
{
    int Age ;
    scanf("%d", &Age );
```

Age >= 18 ? printf("eligible") : printf("You are not eligible & will be eligible after %d years", 18-Age);

}

Que - WAP to input three no.s & print the largest no. using c. op.

```
#include<stdio.h>
int main()
{
    int A, B, C;
    scanf("%d %d %d", &A, &B, &C);
```

L = X>Y ? X : Y

L = Z>L ? Z : L

[6] Bitwise Operator [<<, >>, &, |, ^, ~]

i- Bitwise AND [&]

```
#include<stdio.h>
int main()
{
    char A=10, B=13, C;
        C = A&B;
    printf("%d", C);
}
```

Output → 8

S	64	32	16	8	4	2	1
	0	0	0	1	0	1	0
	0	0	0	1	1	0	1
	0	0	0	1	0	0	0
				✓			

ii- Bitwise OR [|]

```
#include<stdio.h>
int main()
{
    char A=10, B=13, C;
        C = A|B;
    printf("%d", C);
}
```

Output = 15.

S	64	32	16	8	4	2	1
0	0	0	0	1	0	1	0
0	0	0	0	1	1	0	1
0	0	0	0	1	1	1	1
				✓	✓	✓	✓

Age >= 18 ? printf("eligible") : printf("You are not eligible & will be eligible after %d years", 18-Age);

}

Que - WAP to input three no.s & print the largest no. using c. op.

```
#include<stdio.h>
int main()
{
    int A, B, C;
    scanf("%d %d %d", &A, &B, &C);
```

L = X>Y ? X : Y

L = Z>L ? Z : L

[6] Bitwise Operator (<<, >>, &, |, ^, ~)

i- Bitwise AND (&)

```
#include<stdio.h>
int main()
{
    char A=10, B=13, C;
        C = A&B;
    printf("%d", C);
}
```

Output → 8

S	64	32	16	8	4	2	1
	0	0	0	1	0	1	0
	0	0	0	1	1	0	1
	0	0	0	1	0	0	0
				✓			

ii- Bitwise OR (|)

```
#include<stdio.h>
int main()
{
    char A=10, B=13, C;
        C = A|B;
    printf("%d", C);
}
```

Output = 15.

S	64	32	16	8	4	2	1
	0	0	0	0	1	0	1	0
	0	0	0	0	1	1	0	1
	0	0	0	0	1	1	1	1
					✓	✓	✓	✓

Speed
PAGE NO.:
DATE : / /

(iii) Bitwise XOR (^)

A	B	A^B
0	0	0
1	0	1
0	1	1
1	1	0

Same = 0 different = 1

```
#include<stdio.h>
int main()
{
    char A=10, B=13, C;
    C = A^B;
    printf("%d", C);
}
```

Output = 7

iv. Bitwise left Shift (<<)

```
#include<stdio.h>
int main()
{
    char a=10;
    a = a<<1;
    printf("%d", a);
}
```

* First find binary of a=10
* range of char -128 to 127

sign bit

	S	64	32	16	8	4	2	1
* a=10 →	0	0	0	0	1	0	1	0
* a<<1 →	0	0	0	1	0	1	0	0
* a<<2 →	0	0	1	0	1	0	0	0
* a<<3 →	0	1	0	1	0	0	0	0
* a<<4 →	1	0	1	0	0	0	0	0

2's compliment (for a<<4)

	-	1	0	1	1	1	1	1
								+1
a<<4 →	-	1	1	0	0	0	0	0

Speed
PAGE NO.:
DATE : / /

Output

for a<<1 = 20
for a<<2 = 40
for a<<3 = 80
for a<<4 = -96

Ex-

```
#include<stdio.h>
int main()
{
    char a=12;
    a = a<<4
    printf("%d", a);
}
```

	S	64	32	16	8	4	2	1
(a=12)	0	0	0	0	1	1	0	0
(a<<4)	0	1	0	0	0	0	0	0

2's compliment

	-	0	1	1	1	1	1	1
								+1
	-	1	0	0	0	0	0	0

output = -64

NOTE:- N^N° = O.N. * 2^Bits
(New Number, old number)

Que. WAP to multiply a no. by 64 without using multiply operator

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

ex- int x=200;
output = 12800

2^bits = 64
2^bits = 2^6
bits = 6

Speed
PAGE NO.:
DATE: / /

Que. WAP to multiply a no. by 65 without multiply operator.

$2*65 = 130 \rightarrow 2*64+2$

```
#include<stdio.h>
int main()
{
  int x;
  scanf("%d",&x);
  x=(x<<6)+x;
  printf("%d",x);
}
```

V- Bitwise Right shift [>>]

```
#include<stdio.h>
int main()
{
  int x=10;
  x=x>>1;
  printf("%d", x);
}
```

	S	64	32	16	8	4	2	1
x=10 →	0	0	0	0	1	0	1	0
x>>1 →	0	0	0	0	0	1	0	1
x>>2 →	0	0	0	0	0	0	1	0
x>>3 →	0	0	0	0	0	0	0	1
x>>4 →	0	0	0	0	0	0	0	0

Output

x>>1 → 5
x>>2 → 2
x>>3 → 1
x>>4 → 0

Note: NN = ON / 2^bits.

*10/2^1 = 5 *10/2^3 = 1
*10/2^2 = 2 *10/2^4 = 0

V- Bitwise Not [~] [unary]

Ex-

```
#include<stdio.h>
int main()
{
  int x=10;
  x=~x;
  printf("%d",x);
}
```

x=10 →

S	64	32	16	8	4	2	1
0	0	0	0	1	0	1	0

1's compliment

1	1	1	1	0	1	0	1
−	0	0	0	1	0	1	0
							+1
−	0	0	0	1	0	1	1

Output
-11

Ex-

```
#include<stdio.h>
int main()
{
  int x=-10;
  x=~x;
  printf("%d",x);
}
```

(x=10)

S	64	32	16	8	4	2	1
0	0	0	0	1	0	1	0

2's compliment

1	1	1	1	0	1	0	1
							+1

Output:

(x=-10)

1	1	1	1	0	1	1	0
0	0	0	0	1	0	0	1

9

Input Output
Note: N = −(N+1)

Note: For all Bitwise, the input should belong only to char & int family.

Note: -ve no.s are stored in the 2's complement form in the memory.

Ex-
```
#include <stdio.h>
int main ()
{
  char x='A'; //65*2^1=130
  x= x<<1;
  printf("%d",x);
}
```
Output
-126

Ex-
```
#include <stdio.h>
int main ()
{
  char x='0'; //48*2^1=96
  x= x<<1;
  printf("%d",x);
}
```
Output
96

[7] sizeof operator.

* This is used to evaluate the size of datatype.
* sizeof is only one operator which is also a keyword.
  ↳ +ve output.

```
#include <stdio.h>
int main ()
{
  printf("%lu", sizeof(char));
  printf("%lu", sizeof(int));
  printf("%lu", sizeof(short int));
  printf("%lu", sizeof(long int));
  printf("%lu", sizeof(long long int));
  printf("%lu", sizeof(float));
  printf("%lu", sizeof(double));
  printf("%lu", sizeof(long double));
  printf("%lu", sizeof(void));
}
```
Output
1
4
2
4
8
4
8
16
1

* Floating point size by default is double
* double size is 8

Note:- Sizeof is not a function it is a operator.

Ex-
```
#include <stdio.h>
int main ()
{
  printf("%lu\n", sizeof 'A');   → char
  printf("%lu\n", sizeof 2);     → int
  printf("%lu\n", sizeof (12.3)); → float
}
```
Output
4
4
8

Ex-
```
#include <studio.h>
int main ()
{
  int x=5;
  printf("%d %d", sizeof (x=x+10), x);
}
```
Output
4 5.

* Check L value only.
* expression will not solve in sizeof.
* output will depend only on L value in sizeof op.

Ex-
```
#include<stdio.h>
int main()
{
  int x=5;
  float y=3.4;
  double z=4.5;
  printf("%d %d", sizeof(x+y*z), x);
}
```
↳ largest size

Output
8 5

Ex-
```
#include<stdio.h>
int main()
{
  int x = printf("Hello");
  printf("%d", x);
}
```

Output
Hello5

Ex-
```
#include<stdio.h>
int main()
{
  int x = sizeof(printf("%Hello"));
  printf("%d", x);
}
```

Output : 4

String
Collection of characters ended by null character. [\0]

Ex-
```
#include<stdio.h>
int main()
{
  int x = sizeof("Hello");
  printf("%d", x);
}
```
(string ↑)

Output
6

Ex.
```
#include<stdio.h>
int main()
{
  int x = sizeof("Hello World");
  printf("%d", x);
}
```

Output
11

Ex*
```
#include<stdio.h>
int main()
{
  int x = sizeof("Hello\0World");
  printf("%d", x);
}
```

Output : 12.

[B] Comma Operator [,] [left to Right]

* least priority operator in precedence table.
* It separate two expressions.

Ex-
```
#include<stdio.h>
int main()
{
  int x=2,3;
  printf("%d", x);
}
```

Output:
Error (expected identifier)

Ex-
```
#include<stdio.h>
int main()
{
  int x;
  x=2,3;
  printf("%d",x);
}
```

Output:
2.

Ex-
```
#include<stdio.h>
int main()
{
  int x;
  x=(2,3);
  printf("%d", x);
}
```

If paranthesis are used, then the output will be the last value
If not, then the first value.

Output
3

Ex-
```
#include<stdio.h>
int main()
{
  int x;
  x=(printf("Hello"),printf("Hi"));
  printf("%d", x);
}
```

x = (5,2)
HelloHi2

Output:
HelloHi2

Ex-
```
#include<stdio.h>
int main()
{
  int x;
  x=printf("Hello"),printf("Hi");
  printf("%d", x);
}
```

x = 5,2

Output
HelloHi5

a = a + b * c = 5 + 7 * 8 = 61
b = b - c + d = 7 - 8 + 3 = 2
c = (a + d) = 8 * 3 + 61 = 85
d = 3
e = 85

Ex.
```
#include<stdio.h>
int main()
{
  int a=5, b=7, c=8, d=3, e;
  e = (a=a+b*c, b=b-c+d, c=c*d+a);
  printf("%d %d %d %d %d", a, b, c, d, e);
}
```

Output
61 2 85 3 85

[9] Assignment Operators & Short Hand Assignment Operators
* It is used to assign R value expression output into L value variable
* Right to left

[= , += , -= , /= , *= , %= , &= , |= , etc.]

Ex-
```
#include<stdio.h>
int main()
{
  int x;
  x=5;
  printf("%d", x);
}
```

Output
5

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

```
  int x;
  5=x;
  printf("%d", x);
}
```

Output
Error (L-value required)

Note: * L value will always be a variable name.
* R value could be any expression, constant, anything

Ex-
```
#include<stdio.h>
int main()
{
  int x, y;
  x=y=5;
  printf("%d %d", x, y);
}
```

Output
5 5

Ex-
```
#include<stdio.h>
int main()
{
  int a=5, b=7, c=8, d=3, e;
  e = a = a+b*c, b=b-c+d, d=c*d+a;
  printf("%d %d %d %d %d", a, b, c, d, e);
}
```

Output: 61 2 85 3 61

Ex.
```
#include<stdio.h>
int main()
{
  int x=5;
  x+=4;   // x=x+4
  printf("%d", x);
}
```
x-=4 → 1
x*=4 → 20

Output
9

Ex.
```
#include<stdio.h>
int main()
{
  int x=5, y=3;
  x*=y+7;
  printf("%d %d", x, y);
}
```
x*=y+7
x*=3+7
x*=10
x=x*10 = 5*10
x=50, y=10

Output
50 10

* x*=5=y → Not possible (error, L value req.)
* x=5=y → Not Possible
* x+y=10 → Error (L value cannot be an expression)
* z=x+y=10 → Error
* x*=5+7 → Not possible → error

Ex.
```
#include<stdio.h>
int main()
{
  int x=10;
  x&=1;
  printf("%d", x);
}
```
x&=1
x=x&1
1010 → 10
0001 → 1
0000

Output
0

Ex.
```
#include<stdio.h>
int main()
{
  int x=10, y=13;
  x&=y;
  printf("%d", x);
}
```
x&=y
x=x&y
1010 → 10
1100 → 13
1000
8

Output
8

[10] Unary plus and Unary minus

Unary minus

```
#include<stdio.h>
int main()
{
  int x=10, y;
  y=-x;
  printf("%d", x);
}
```

Output
-10

* It changes the sign.

Ex- { int x = 10, y ;
-x = y ;
printf("%d", x);
}

Output:
Error (L value requsid)

* It changes L-value into R-value.

Ex- #include<stdio.h>
int main()
{
int i;
float f;
char c;
short int s;
double d;
printf("%lu\n", sizeof(-i)); Output: 4
" (-f)); 4
" (-c)); 4
" (-s)); 4
" (-d)); 8
}

* It is used for integer promotion.

Unary Plus

{
int x=10, y;
y = +x;
printf("%d", y);
}

Output
10

{
int x=-10, y;
y = +x;
printf("%d", y);
}

Output:
-10

* It changes L-value into R-value.
* It is used for integer promotion.

[II] Increment & Decrement operator
i- Post inc. (p++) → first uses value then inc. by 1.
ii- Pre inc. (++p) → first inc. by 1 then uses the value.
iii- Post dec. (p--) → first uses value then dec. by 1.
iv- Pre dec. (--p) → first dec. by 1 then uses the value.

Ex- #include<stdio.h>
int main()
{
int x=10, y;
y = x++;
printf("%d %d", x, y);
}

Output
11, 10.

Ex-
```
#include<stdio.h>
int main()
{
    int x=10, y;
    y= ++x;
    printf("%d, %d", x, y);
}
```

Output:
11, 11.

Ex-
```
#include<stdio.h>
int main()
{
    int x=10, y;
    y= x--;
    printf("%d, %d", x, y);
}
```

Output
9, 10

Ex-
```
#include<stdio.h>
int main()
{
    int x=10, y;
    y= --x;
    printf("%d, %d", x, y);
}
```

Output:
9, 9

Ex-
```
#include<stdio.h>
int main()
{
    int x=10, y;
    y= 10++;
    printf("%d, %d", x, y);
}
```

Output:
Error (lvalue required)

Ex-
```
#include<stdio.h>
int main()
{
    int x=5, y=7, z=6, A;
    A= x++ * --y / ++z;
    printf("%d, %d, %d, %d", x, y, z, A);
}
```

p++, p-- → L-R
++p, --p → R-L

6 6 7

5 * 6 / 7
30 / 7
5

~~5*1 = 5~~

Output:
6 6 7 4

Ex-
```
#include<stdio.h>
int main()
{
    int x=10, y;
    y= ++++x;
    printf("%d, %d", x, y);
}
```

Output:
(l-value req.) Error

Ex. #include <stdio.h>
int main()
{
int x=8, y;
y = x++ * x++;
printf("%d, %d", x, y);
}

Output:
Compiler dependent, Undefined behaviour, will not follow a particular sequence point.

Decision Control Statement :-

- Conditional
  - Simple if
  - if else
  - if else if ladder
  - Nested if else
  - switch case
- Unconditional
  - Go to

1. Simple if

Syntax :-

```
if (condition)
{
    Block B
}
Block C
```

optional if block has just one statement

Ex.
```
#include <stdio.h>
int main()
{
    int x=10;
    printf("%d", "Hi");
    if (x<20)
    {
        printf("Hello");
        x = x+15;
    }
    printf("%d", x);
}
```

Output:
HiHello25

Ex.
```
#include <stdio.h>
int main()
{
    int x=100;
    printf("Hi");
    if (x<20)
    {
        printf("Hello")
        x= x+15
    }
    printf("%d", x);
}
```

Output:
Hi115

Ex-

```
#include <stdio.h>
int main()
{
  int x=10;
  printf("Hi");
  if (x<20)
  printf("Hello");
    x=x+15
  printf("%d", x);
}
```

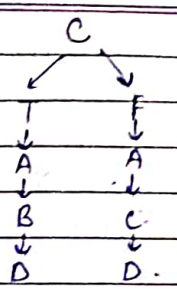
Output:
HiHello25.

Ex-

```
#include <stdio.h>
int main()
{
   int x=10;                    x=100
  printf("Hi");
  if (x<20);
  printf("Hello");
     x=x+15;
  printf("%d" x);
}
```

Output:
HiHello25. HiHello115

2. if else



Syntax :-

```
Block A
if (condition)
{
   Block B
}
else
{
   Block C
}
   Block D
```

Ex-

```
#include <stdio.h>
int main()
{
  int x= 7 ;   17
  printf("Hi");
  if (x<10)
  {
    printf("Hello");
      x=x+19;
  }
  else                          HiBye13
  {
    printf("Bye");
      x=x-4;
  }
  printf("%d", x);
}
```

Output:
HiHello26.

Ex -
```
#include <stdio.h>
int main()
{
  int x=7;
  printf("Hi");
  if (x<10)
  { printf("Hello")
    x=x+19
  }
  else
    printf("Bye");}
    x=x-4;
  printf("%d", x);
}
```
Output:
HiHello22.

Ex -
```
#include <stdio.h>
int main()
{
  int x=7;
  printf("Hi");
  if (x<10)
  printf("Hello");
    x=x+19;
  else
  printf("Bye");
    x=x-4;
  printf("%d", x);
}
```
Output:
Error
[else without if]

Ex -
```
#include <stdio.h>
int main()
{
  int x=7;
  printf
  if (x<10)
    printf("Hello");
  else
    printf("Bye");
    x=x-4;
  printf("%d", x);
}
```
else;
HiHelloBye3

Output
HiHello3.

Que - WAP to input a no. & check no. is even or odd :-

```
#include <stdio.h>
int main()
{  int X;
  printf("Enter a number");
  scanf("%d", &X);
  if X%2==0
    printf("Number is even")
  else
    printf("NO. is odd");
}
```

N - (N/2)*2==0
(N/2)*2==N
(N>>1)<<1==N
(N&1)==0

Que- WAP to input a year & check if it leap year or not

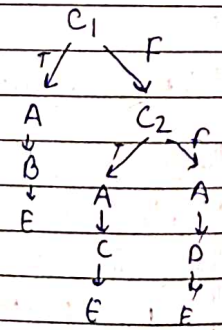
```
#include <stdio.h>
int main ()
{
  int x;
  if (x%400==0 || x%4==0 && x%100!=0)
    printf("leap year")
  else
    printf(" Not leap year")
}
```

3. if else if ladder.

Syntax:-

```
Block A
if (C1)
{
    Block B
}
else if (C2)
{
    Block C
}
else
{
    Block D
}
Block E
```

optional



Ex-

```
#include<stdio.h>
int main ()
{
  int x=10;
  printf("Hi");
  if (x<20)
  { printf("Hello")
    x=x+70;
  }
  else if (x==80)        ← without else.
  { printf("Hey")
    x=x+9
  }
  else
  { printf("Bye")
    x=x-17;
  }
    printf("%d",x);
}
```

Output
HiHelloBO.

Ex-

```
#include <stdio.h>
int main ()
{
  int x=100;
  printf("Hi");
  if (x<20)
  {
    printf("Hello")
    x=x+70;
```

```
    else if (x==80)
    { printf("Hey");
      x=(x+9)
    }
    printf("%d",x);
}
```

Output
Hi 100.

```
else
{ print "Bye");
  x = x-7;
}
```
→ Hi Bye 93.

Ex-
```
#include<stdio.h>
int main ()
{
  int x=100;
  printf("Hi");
  if (x<20)
  {
    printf("Hello")
      x=x+70
  }
  else if (x==80)
  {
    printf("Hey")
      x=x+9
  }
  else
  {
    printf("Hey")
      x= x-73
  }
  printf("%d",x);
}
```

Output
Hi Bye 93

Ex-
```
#include<stdio.h>
int main ()
{
  int x=10;
  printf("Hi");
  if (x<20
  {
    printf("Hello")
      x=x+70;
  }
  else if (x==80)
  { printf("Hey")
      x=x+9;
  }
  else ;
    printf("Bye");
      x=x-17;
  printf("%d",x);
}
```

Output → HiHello Bye63.

Que- WAP to input a no. & check no. is +ve, -ve & 0.

```
#include<stdio.h>
int main ()
{
  int x;
  printf("Enter a Number");
  scanf("%d",&x);
```

```
if (x>0)
{
  printf("Number is positive");
}
else if (x<0)
{
  printf("Number is negative")
}
else
{ printf("Number is zero")
}
}
```

Que- WAP to input the age of a person and check if he is eligible for voting or not. If not check when he will be eligible.

```
#include <stdio.h>
int main ()
{
  float x,a;
  printf("Enter your Age");
  scanf("%f", &x);
  if (x>17)
  {
  printf("You are eligible for voting");
  }
  else
  {
  printf("You will be eligible after");
    a=18-x;
    printf("%.1f years", a);
  }
}
```

Que- WAP to input two numbers & find which one is greater.

```
#include <stdio.h>
int main ()
{
  int a,b;
  printf("Enter two numbers");
  scanf("%d %d", &a, &b);
  if (a>b)
  {
  printf("a is greater");
  }
  else if (b>a)
  {
  printf("b is greater");
  }
  else
  {
  printf("Numbers are equal");
  }
}
```

4. Nested if else

Syntax :-

```
if (C1)
{
    if (C2)
    {
        Block A
    }
}
```

Ex -

```
#include <stdio.h>
int main()
{
    int x=40;                    n = 0
    printf("Hi");
    if (x<<2)
    {
        printf("%d", x);
        if (x=x<<1)
        {
            printf("Hello");
            printf("%d", x);
        }
    }
else
    {   if (x==0)
        {
            printf("Hey");
            x=x+7;
        }
```

```
        else
        {   printf("Section A");
            x= x%9;
        }
        printf("Bye");
    }
    printf("%d", x);
}
```

Output:

Hi40HelloBOBO

Ex -

```
#include <stdio.h>
int main()
{
    int x=0;
    printf("Hi");
    if (x<<2)
    {   printf("%d", x);
        if (x=x<<1){
            {   printf("Hello");
                printf("%d", x);
            }
    }
else
    {   if (x==0)
        {   printf("Hey");
            x = x+7;
        }
```

```
        else
        { printf("Section A");
            x = x % 9;
        }
        printf("Bye");
    }
    printf("%d", x);
}
```

Output

HiHeyBye 7

Ex-

```
#include <stdio.h>
int main()
{
    int x = -5;
    printf("Hi");
    if (x<<2)
    {
        printf("%d", x);
        if (x=x<<1)
        {
            printf("Hello");
            printf("%d", x);
        }
    }
    else
    { if (x==0)
        { printf("Hey");
            x = x+7;
        }
```

```
    else
    { printf("Section A");
        x = x % 9;
    }
    printf("Bye");
}
printf("%d", x);
}
```

Output

Hi-5Hello-10-10

Ex-

```
#include <stdio.h>
int main()
{
    int x = 0;
    printf("Hi");
    if (x<<2)
    { printf("%d", x);
        if (x = x<<1)
        { printf("Hello");
            printf("%d", x);
        }
    }
    else ;
    { printf("Section A");
        x = x % 9;
    }
    printf("Bye");
}
printf("%d", x);
```

```
if (x==0)
{ printf("Hey");
    x = x+7;
}
else;
```

WAP to in. the coordinates of a point of a plane & check the Quadrant.

Output

Hi Hey Section A Bye 7.

Ex-

```
#include <stdio.h>
int main()
{
  int x=10;
  printf("Hi");
  if (x<<2);
  {   printf("%d",x);
      if (x=x<<1)
      {
        printf("Hello");
        printf("%d",x);
      }
  }
else;
    { if (x==0)
        { printf("Hey");
          x=x+7;
        }
      else
        { printf("Section A");
          x=x%9;
        }
      printf("Bye");
    }
  printf("%d",x);
}
```

Output: Error (Else without previous if)

9

Que- WAP to in. the coordinates of a point of a plane & check the quadrant.

```
#include <stdio.h>
int main()
{
  int x,y;
  scanf("%d %d",&x,&y);
    if (x>0)
    {
      if (y>0)
      {
        printf("1st Quadrant");
      }
      else if (y<0)
      {
        printf("4th Quadrant");
      }
      else
      {
        printf("On x-axis");
      }
    }
    else if (x<0)
    {
      if (y>0)
      { printf("2nd Quadrant"); }
      else if (y<0)
      {
        printf("3rd Quadrant");
      }
```

if (x>0)
{ if (y>0)
1st
else if (y<0)
4th
else
on x axis
}
else if x<0
{ if (y>0)
2nd
else if (y<0)
3rd
else
on -ve x axis
}
else if (y>0)
on +ve y axis
else if (y<0)
on -ve y axis
else
on origin

WAP to in three no: & find largest among them using nested

```
        else
          {
            printf("on -ve x-axis");
          }
      }
    else
     { if (y>0)
        {
          printf("on +ve y-axis");
        }
        else if (y<0)
          {
            printf("on -ve y-axis");
          }
        else
          {
            printf("on origin");
          }
      }
}
```

Ques- WAP to three no.s & find largest among them using nested.

```
#include <stdio.h>
int main ()
{
 int a, b, c;
 scanf("%d %d %d", &a, &b, &c);
    if (a>b)
      {
```



```
    if (a>=b)
      {   printf("%d", a);
      }
    else
      {  printf("%d", c);
      }
else if (b>=a)
    {
      if (b>=c)
        {
          printf("%d", b);
        }
      else
        {
          printf("%d", c);
        }
}
```

Operator	Description	Associativity
() [] . ->>> ++ --	Parentheses or function call Brackets or array subscript Dot or Member selection operator Arrow operator Postfix increment/decrement	left to right
++ -- + - ! ~ (type) * & sizeof	Prefix increment/decrement Unary plus and minus not operator and bitwise complement type cast Indirection or dereference operator Address of operator Determine size in bytes	right to left
* / %	Multiplication, division and modulus	left to right
+ -	Addition and subtraction	left to right
<< >>	Bitwise left shift and right shift	left to right
< <= > >=	relational less than/less than equal to relational greater than/greater than or equal to	left to right
== !=	Relational equal to and not equal to	left to right
&	Bitwise AND	left to right
^	Bitwise exclusive OR	left to right
|	Bitwise inclusive OR	left to right
&&	Logical AND	left to right
||	Logical OR	left to right
? :	Ternary operator	right to left
= += -= *= /= %= &= ^= |= <<= >>=	Assignment operator Addition/subtraction assignment Multiplication/division assignment Modulus and bitwise assignment Bitwise exclusive/inclusive OR assignment	right to left
,	Comma operator	left to right

Operator Precedence and Associativity

Speed
PAGE NO.:
DATE : / /

[5] Switch case

Syntax : switch (Expression) → int/char
{
case value1 : Block 1
break;
case value 2 : Block 2
break;
case value n : Block N
break;
default : default block
}

expression can be used but with constant like 1*1.

optional ← block

Ex- WAP to input a day no. from 1 to 7 & print the corresponding day name using switch case.

```
#include <stdio.h>
int main ()
{
    int d;
    printf("Enter a day number from 1 to 7");
    scanf("%d", &d);
    switch (d)
        case 1 : printf("Monday");
                 break;
        case 2 : printf("Tuesday");
                 break;
```

Case label ← Case 2.0
does not reduce to an integer constant

```
case 3 : printf("Wednesday");
         break;
case 4 : printf("Thursday");
         break;
case 5 : printf("Friday");
         break;
case 6 : printf("Saturday");
         break;
case 7 : printf("Sunday");
         break;
default : printf("Invalid day no.");
}
```

Note:
* Default block is optional in switch case.
* Break keyword is mandatory after every case body otherwise it will keep on executing all the case bodies until break will encounter.
* If default block is placed in the beginning or in b/w any of case then we have to use break after default block.
* Switch quantity or expression should always provide an integer value.
* Switch label or value can be an expression of int or char. constants.
* Case label should be only integer or character constant only.
* Case label/value cannot be a variable.
* All the case label should be unique

Loop Control Statement:

While, for, do while
- entry control loop / pre-test
- exit control / post test

(1) while

Syntax:
```
Step1 ← Initialisation of loop variable;
while (condition)        condition is mandatory
{                  Step 2
   if true ✓
   Body
Step3 Updation of loop variables   increase/decrease
}
```

Ex- WAP to display your name 5 times.

```
#include<stdio.h>
int main()
{
   int i=5;                 i=1
   while (i>=1)             i<=5
   {
      printf("Name\n");
      i=i-1;                N=N+1
   }
}
```

display N no. of times
int n
scanf("%d",&n);
while(n>0)
{ printf("Name\n");
n=n-1;
}

Ex- WAP to print first n natural no.

```
#include<stdio.h>
int main()
{
   int i=1,n;
   scanf("%d",&n);
```

```
while (i<= n)
{
    printf("%d\n", i);
    i++;
}
```

for n = 5: 1 2 3 4 5

Ex- WAP to print natural no. in reverse order.

```
#include <stdio.h>
int main()
{
    int n;
    scanf("%d", &n);
    while (n>0)
    {
        printf("%d\n", n);
        n--;
    }
}
```

for n = 5: 5 4 3 2 1

Ex- WAP to print all the even nos from 1 to n

```
#include <stdio.h>
int main()
{
    int i=2, N;
    scanf("%d", &N);
    while (i<=N)
    {
        printf("%d\n", i);
        i=i+2;
    }
}
```

while (i<=N)
{ if(i%2==0)
printf("%d\n", i);
i=i+1;

Ex- WAP to print first n even nos.

```
#include <stdio.h>
int main()
{
    int i=2, N;
    scanf("%d", &N);
    while (i<=2*N)
    {
        printf("%d\n", i);
        i=i+2;
    }
}
```

for odd (i<=2*N-1)

i=2, N;
(N>0)
pf(" ", i)
i=i+2;
N=N-1;

Ex: WAP to print even no. from n to 1.

```
#include <stdio.h>
int main()
{
    int n, i;
    scanf("%d\n", &n);
    if (N%2==0)
        N=N-1;
    while (N>0)
    {
        printf("%d\n", n);
        n=n-2;
    }
}
```

while(N>0)
{ if (N%2==0)
printf("%d\n", N);
N=N-1;
}

Ex- WAP to print the sum of first n natural no.

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

Ex- WAP to find the factorial of a no.

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

Q WAP to input a no. & print all the factors of a no in reverse order

WAP to check whether a no is palindrome or not

Ex- WAP to input a no. & count the no. of digit in that no.

```
#include<stdio.h>
int main()
{
    int N, d=0;
     scanf("%d", &n);
    while (N>0)
     {   d++;
         N=N/10;
     }
     printf("%d\n", d);
}
```

to check prime no

Que- WAP to input no. & print all the factors of a no.

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

Que2- WAP to input no. & print all the -- in reverse order.

Speed
PAGE NO.:
DATE : / /

```
# include <stdio.h>
  int main ()
{
    int i, N;
      i=N;
    scanf ("%d", &N);
     while (i<=N && i>0)
          {
            if (N%i==0)
               printf ("%d\n", i);
                 i--;
          }
}
```

Que- Perfect No.

```
# include <stdio.h>
  int main ()
{
   int N, s, i=1, s=0;
    scanf ("%d" &N);
    while (i<N)
      {
         if (N%i==0)
           s=s+i;
             i++;
      }
```