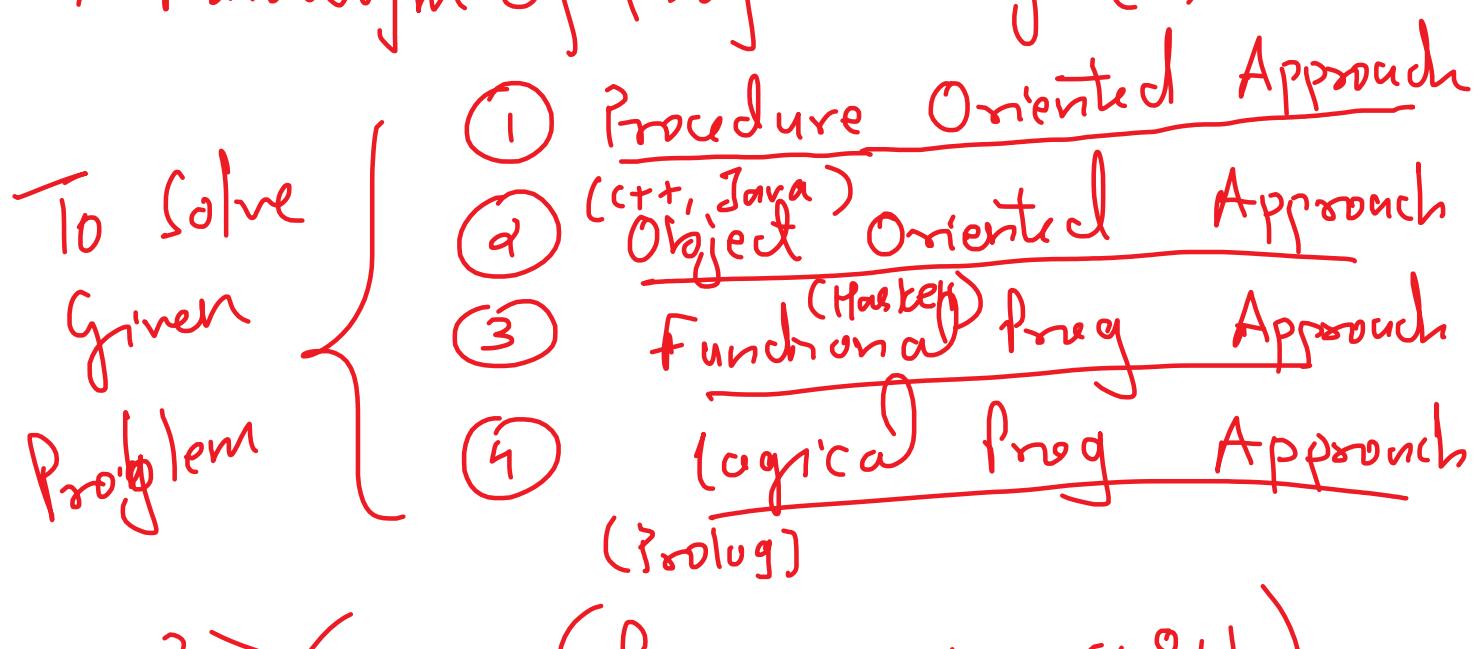


Structured Programming Approach } Kyon?

\* Programming Subject

\* C Programming !!

\* Paradigm of Programming (C)



Structured Programming Approach

Procedure Oriented Programming Approach

\* C language employs structured) Programming Approach

\* Structured Programming Approach

\* (Jab bhi koi problem bataye  
raise a question)

\* Program to find the root of Quadratic Eq<sup>n</sup>

Yaha Question raise hoga  $\Rightarrow$  Kaise??

(Kaise nikalenge root of Eq<sup>n</sup>)  
Matlab formula Kya hai  
Structured  
Prog  
Approach

Ya Computation Based

Jab Question Kaise Hota hai  $\Rightarrow$  Kiska

Jab Uveshon Karne Mota hua  $\Rightarrow$  Kiska

## Object Oriented Approach

(structured)

Procedure Oriented  
Approach

① Question Arises

Kaise How

Computation Based

② Data is not  
important, Procedure  
is important

③ C

Object Oriented  
Approach

① Question Arises

Kiska

Object Based

② Procedure is not  
important, Data is  
important

③ C++, Java

Program  $\rightarrow$  Set of Instructions to accomplish  
a desired task

Program language  $\Rightarrow$  language of Instructions  
Computer Program  $\Rightarrow$  When the Instructions  
are given to Computer/  
Machine

Computer Programming Language  $\rightarrow$  language used  
to give Instructions to Machine

M/C understands  $\rightarrow$  Binary language (0/1)

Faltu Example  $\Rightarrow$  Kaise ho bhai (English)

Binary Main  $\Rightarrow$  011011011011011001

\* If Instructions are Simple, we can write  
into Binary

\* If Instructions are Complex it is difficult to specify Instructions in Binary

\* Joh Machine Ko Samjha woh low level language

\* Joh machine Ko Nahi Samjha woh high level language  
(C, C++, Java, Python)

\* Developers likes high level language  
So Developers will write code in high level language (for eg C.)

\* Lekin Machine Ko high level (i.e C)  
Nahi Samajhta

\* So we need a Connector that converts high level Code to native m/c Code.  
(Done by Compiler)

Kahani ! !

m/c

S/w

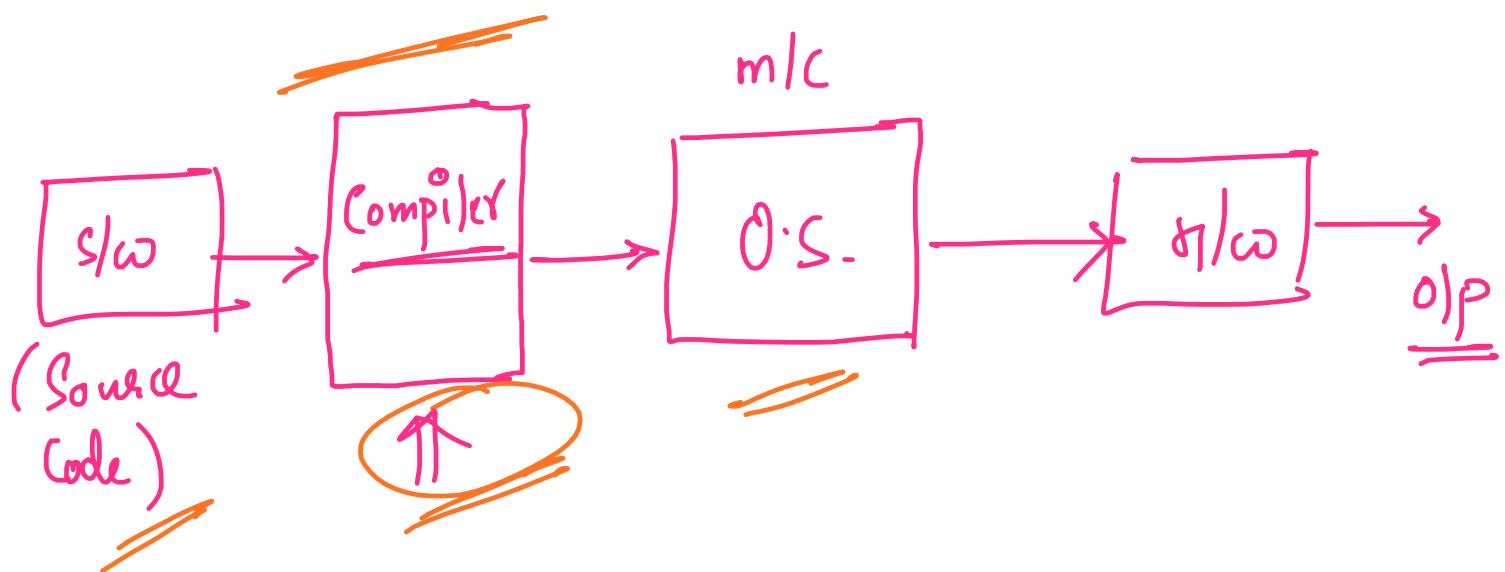
(code)

O.S.

H/w

O.S acts as Interface  
bet^n S/w & H/w

"Code Sabse pehle O.S ko Samajhna chahiye"



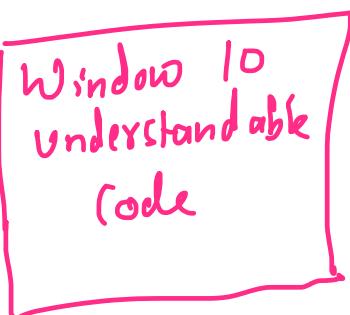
Compiler Converts Source Code to native  
m/c code

(Source Code)

E1.c



Compiler Output



E1.c



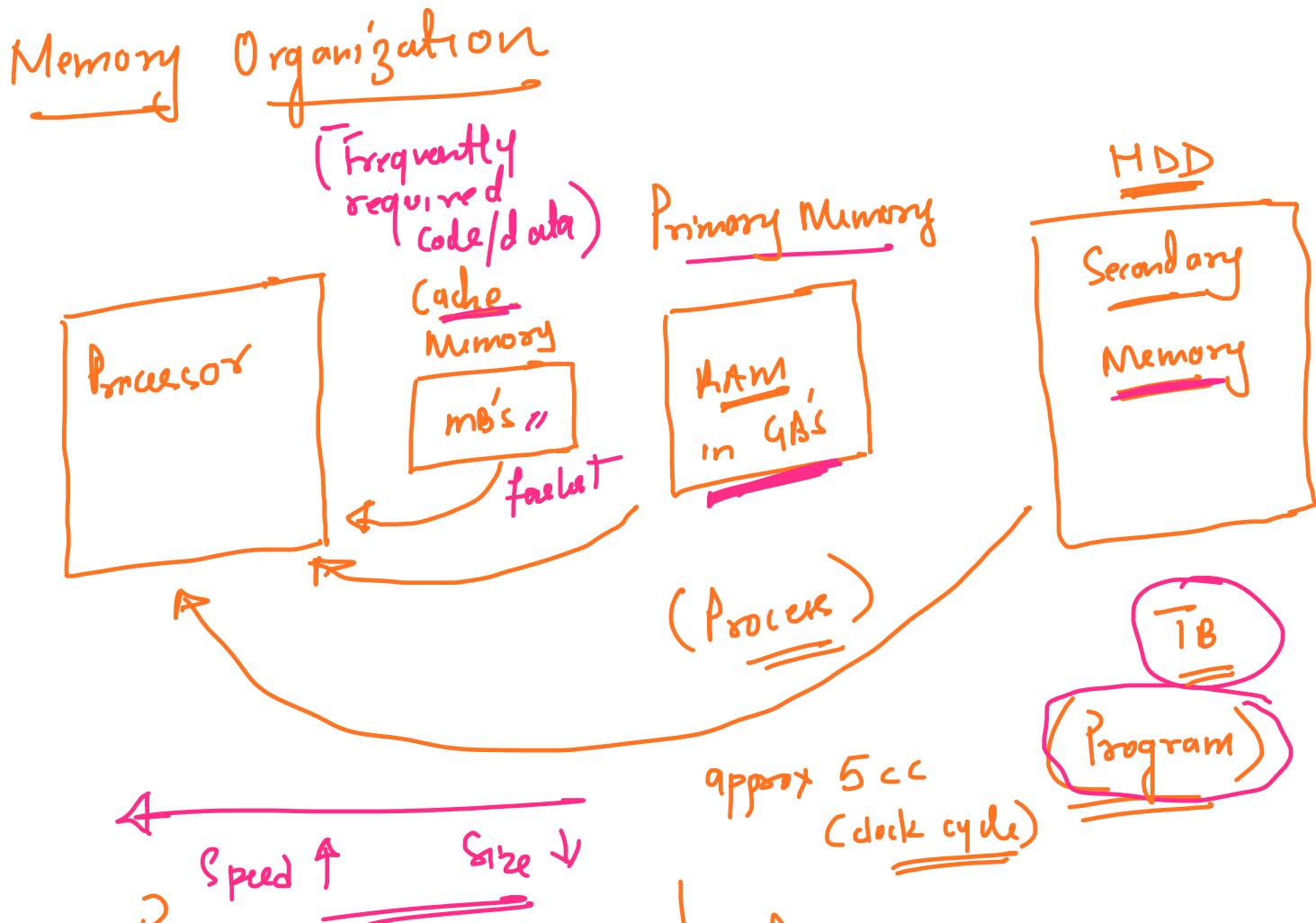
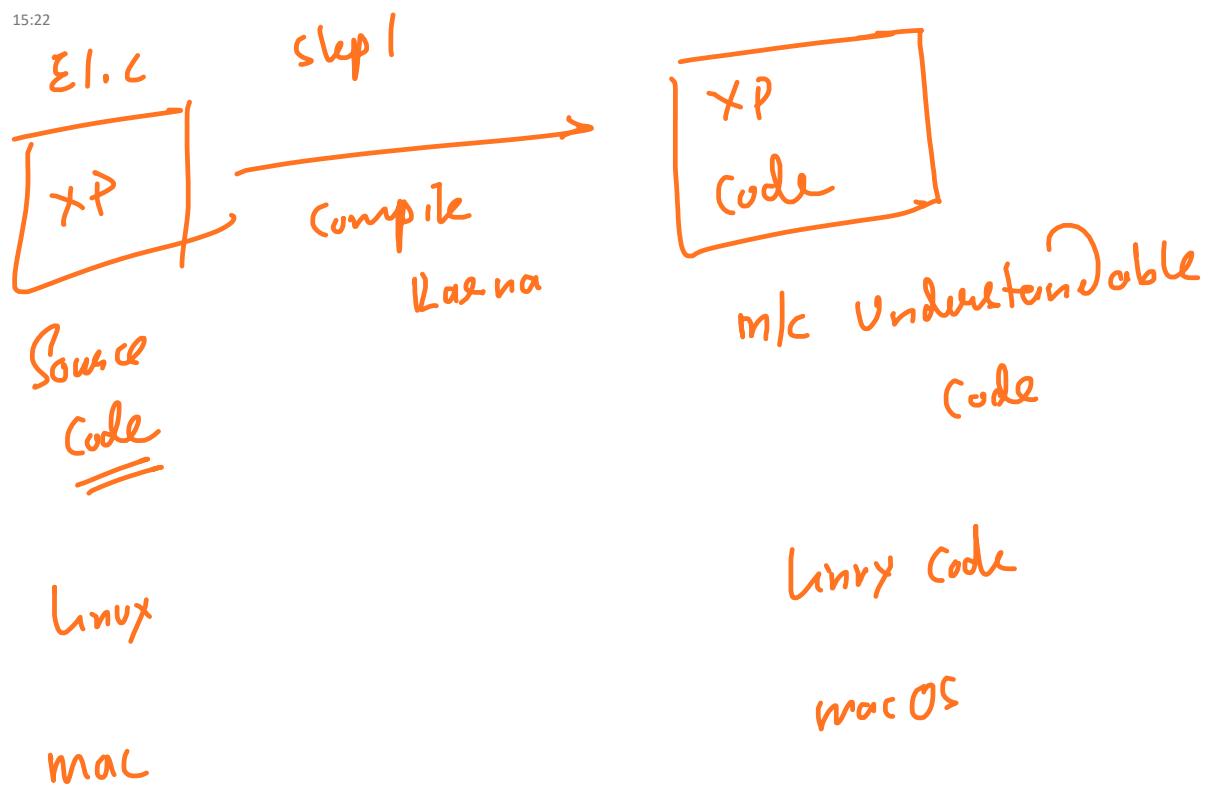
Compiler

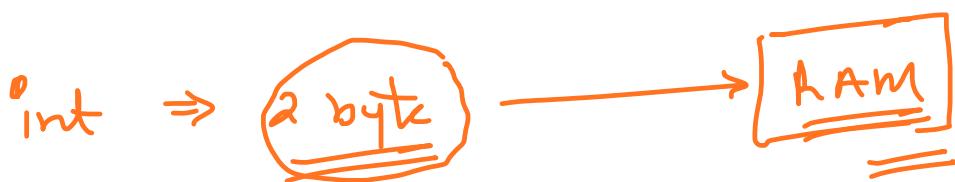
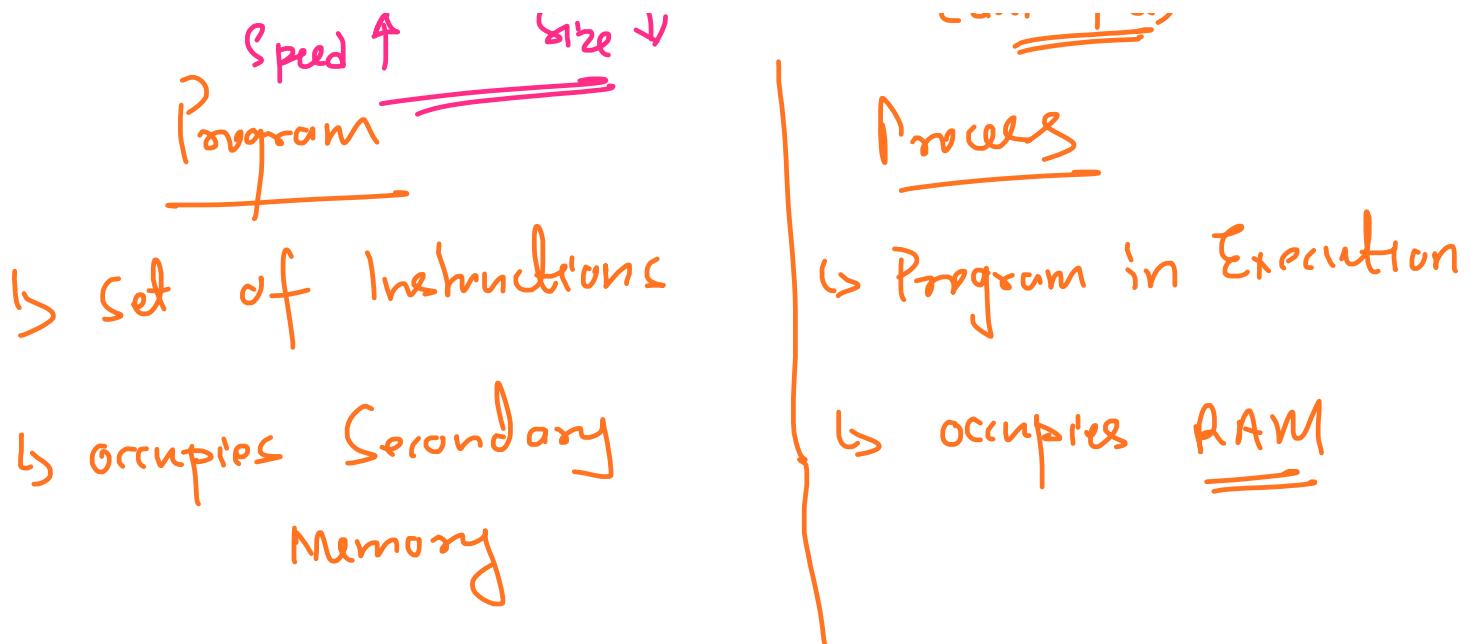


"

native → Jis OS pe compile karoge, woh  
 OS understandable code banega "

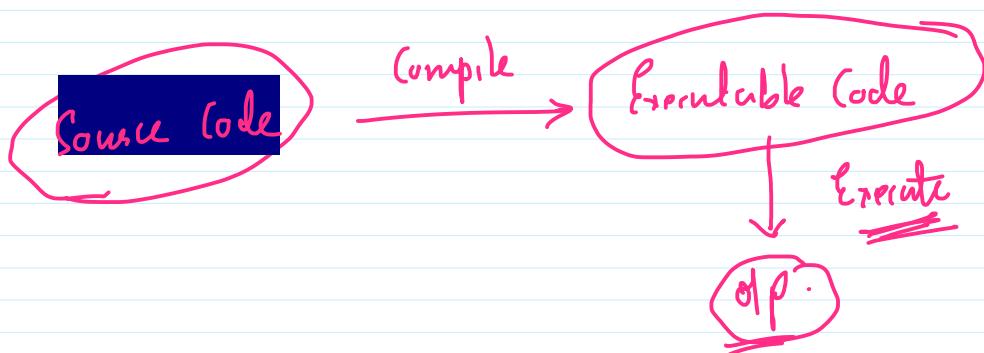
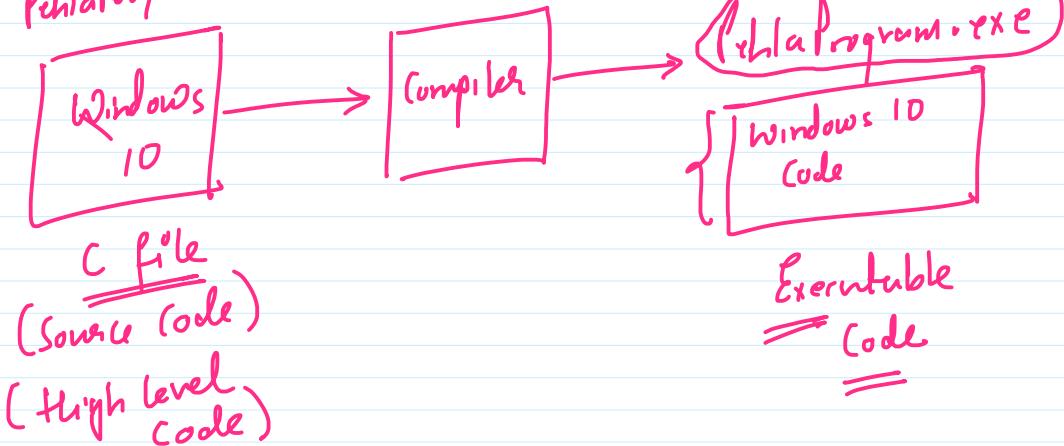
CP - SDW (Sanjeev Dwivedi)





# C Programming

Pehla program . C



## Preprocessor Directives

#include<stdio.h> standard input output header file

Return Type int main() right parenthesis }

Data Type int a,b,sum; left parenthesis

Opening Curly bracket Every statement must end with Semicolon

printf("plz enter two integers\n"); To print on Screen

scanf("%d %d",&a,&b); format specifier

sum=a+b; to read from Input device

Assignment printf("result = %d",sum);

Closing Curly bracket return 0;

}. return Keyword

# include <stdio.h> ⇒ Preprocessor Subce public  
number dimension mean standard

~~#include <stdio.h>~~  $\Rightarrow$  Preprocessor source part  
sample program mein standard  
header file available kar dega

Header file  $\Rightarrow$  Ek jaise kaam karne wale saare  
ready made fn header file main store  
kiye jaate hain

stdio.h  $\leftarrow$  Input and output ka kaam karne  
wale fn

Ex scanf() } Ready made & stored  
printf() } in <stdio.h>

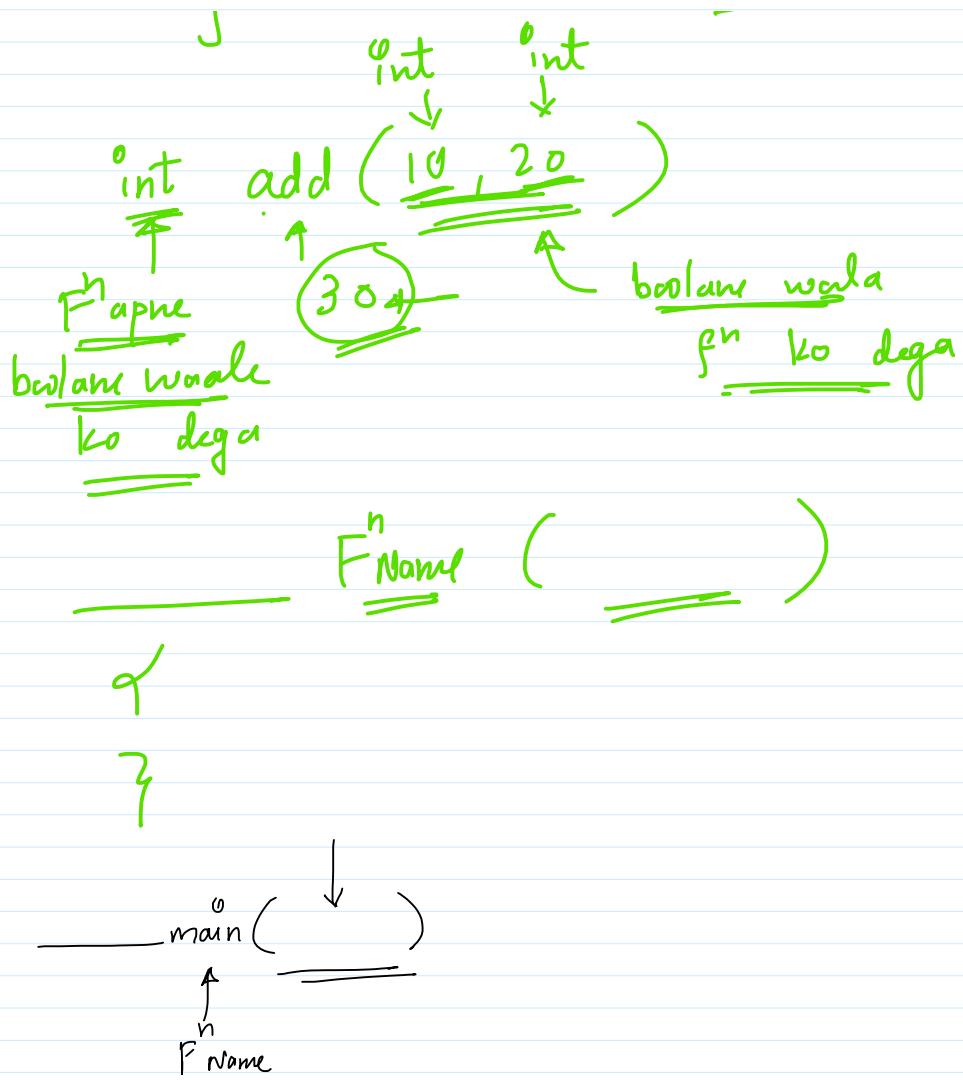
PreProcessor  $\Rightarrow$  It will perform the actions needed  
before the program goes to processor

Ex  $\Rightarrow$  Including the header file.

Note start  $\rightarrow$  {  
main()  
End  $\rightarrow$  } } Scope of loop/  
fn etc.

Note  $\circled{int}$  main( )  $\downarrow$   $\rightarrow$  Yahan pe hum  
woh pass karne  
hain jo main ko  
bulane waala  
data hai

Scope  $\rightarrow$  return  $\circled{0}$   $\Leftarrow$  "All is well"  
} o+ o+



`#include <stdio.h>`  $\Rightarrow$  to use printf and scanf

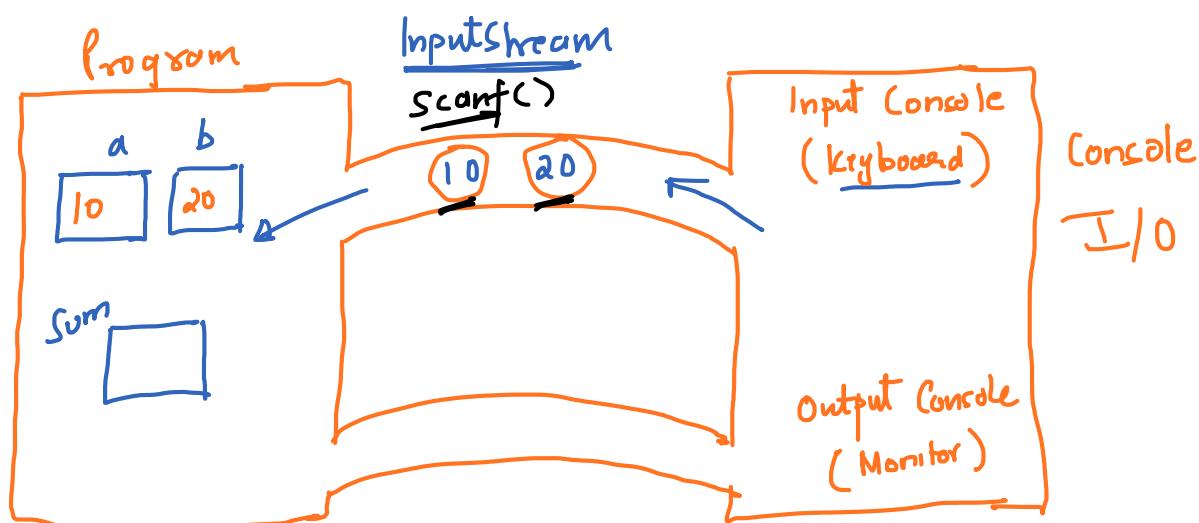
int main()  $\leftarrow$  To call main no arg is required

start  $\rightarrow$  |

```
printf(" Sab Moh maya Hai \n");
printf(" C Programming \n");
```

Escape character  
new line character

end  $\rightarrow$  return 0;  
 $\leftarrow$  Main want to return status that there was no problem during prog Execution.



InputStream  $\Rightarrow$  It brings the input entered by the user into the program

\* The numbers entered by user from Keyboard are in input Stream

`scanf("%d", &a);`  $\Rightarrow$  Read a decimal integer value from InputStream

`scanf( "%d", &a )`  
 format Specifier  
 [ Specifies type of value to read or write ]  
 $\underline{\underline{\%d}}$   $\Rightarrow$  decimal  $\underline{\underline{\text{integer value}}}$   
 $(10, 20, -5, -1, \dots)$

value from Input Stream  
 Extract the value and assign it to address of memory known as a

$\underline{\underline{\%f}}$   $\Rightarrow$  floating point Value  $(10.5, 20.8, 6.3)$  \* Assignment is always done at address

$\underline{\underline{\%c}}$   $\Rightarrow$  character values  $('A', 'a', 'z')$   
 In scanf we must use ampersand (&)

`scanf("%d", &b);`

To read two values.

$\left\{ \begin{array}{l} \text{scanf}(" \%d", \&a) ; \\ \text{scanf}(" \%d", \&b) ; \end{array} \right.$

OR  $\text{scanf}(" \%d \%d", \&a, \&b) ;$

For reading 3 values  $\Rightarrow \text{scanf}(" \%d \%d \%d", \&a, \&b, \&c) ;$

Suppose 2 float values  $\Rightarrow$

$\text{scanf}(" \%f \%f", \&a, \&b) ,$

\* Suppose 1<sup>st</sup> Integer, 2<sup>nd</sup> float, 3<sup>rd</sup> int

~~scanf("%d %f %d", &a, &b, &c);~~

Now printf

printf("Hello");

int a=10;

printf("%d", a);

A decimal int  
value will be  
displayed

and a is name of  
Memory jiska value  
display

⇒ scanf("%d", &a); ⇒ Read int value and assign to a

⇒ printf("%d", a); display int value and it will be  
value of a

\* int a=10, b=20;

printf("a=%d", a); } OR

printf("b=%d", b); } }

printf("a=%d b=%d", a, b);

\* Suppose.

int a=10;

float b=10.5;

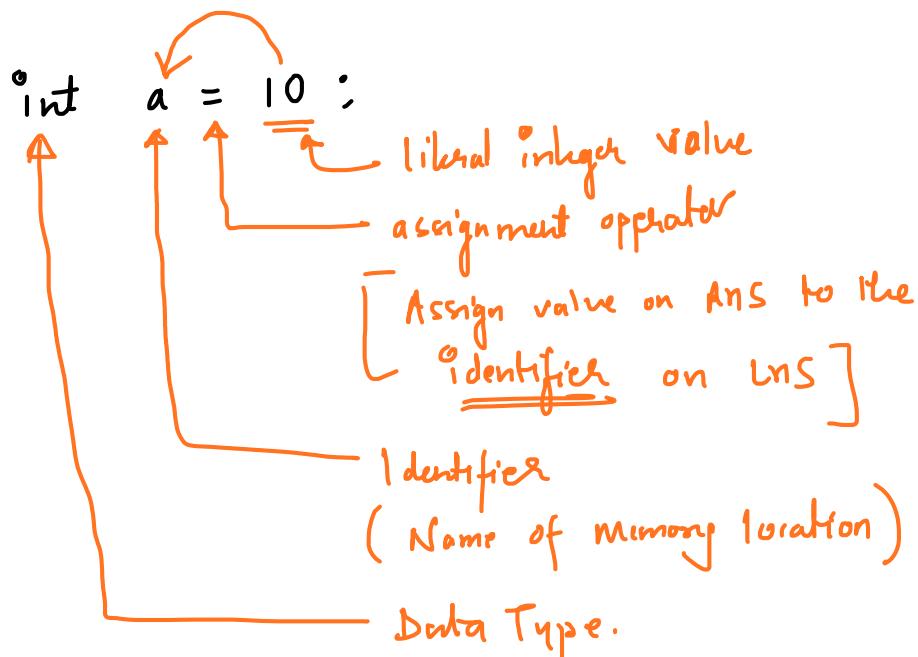
int c=15;

```
printf(" a=%d b=%f c=%d", a, b, c);
```

OR

```
printf(" Value of a = %d \n Value of b = %f \n
Value of c = %d ", a, b, c);
```

Note →



\* Imp

Data Type → To perform any operation on value we must store them in memory

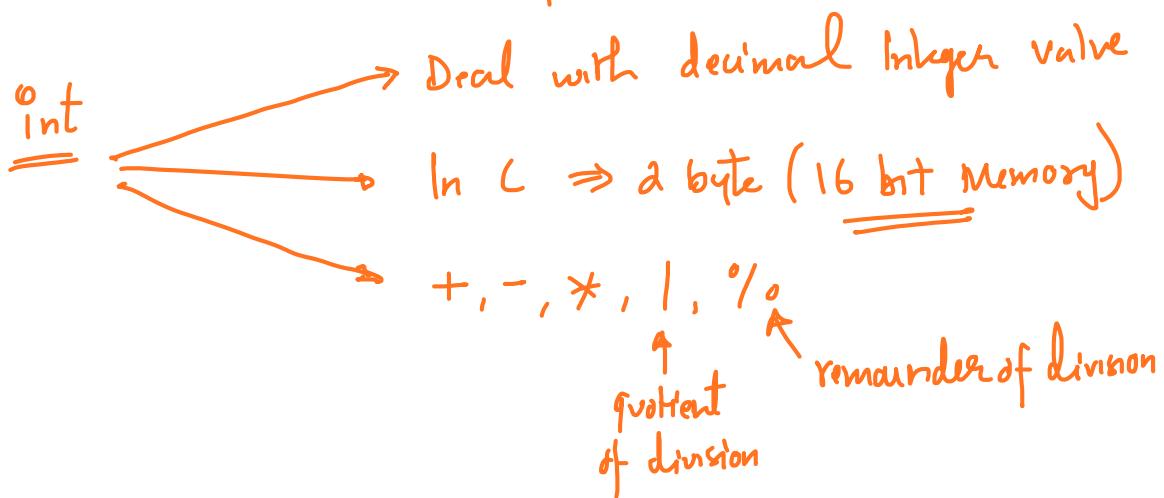
\* Now to access the memory we must assign name to the memory (Identifier)

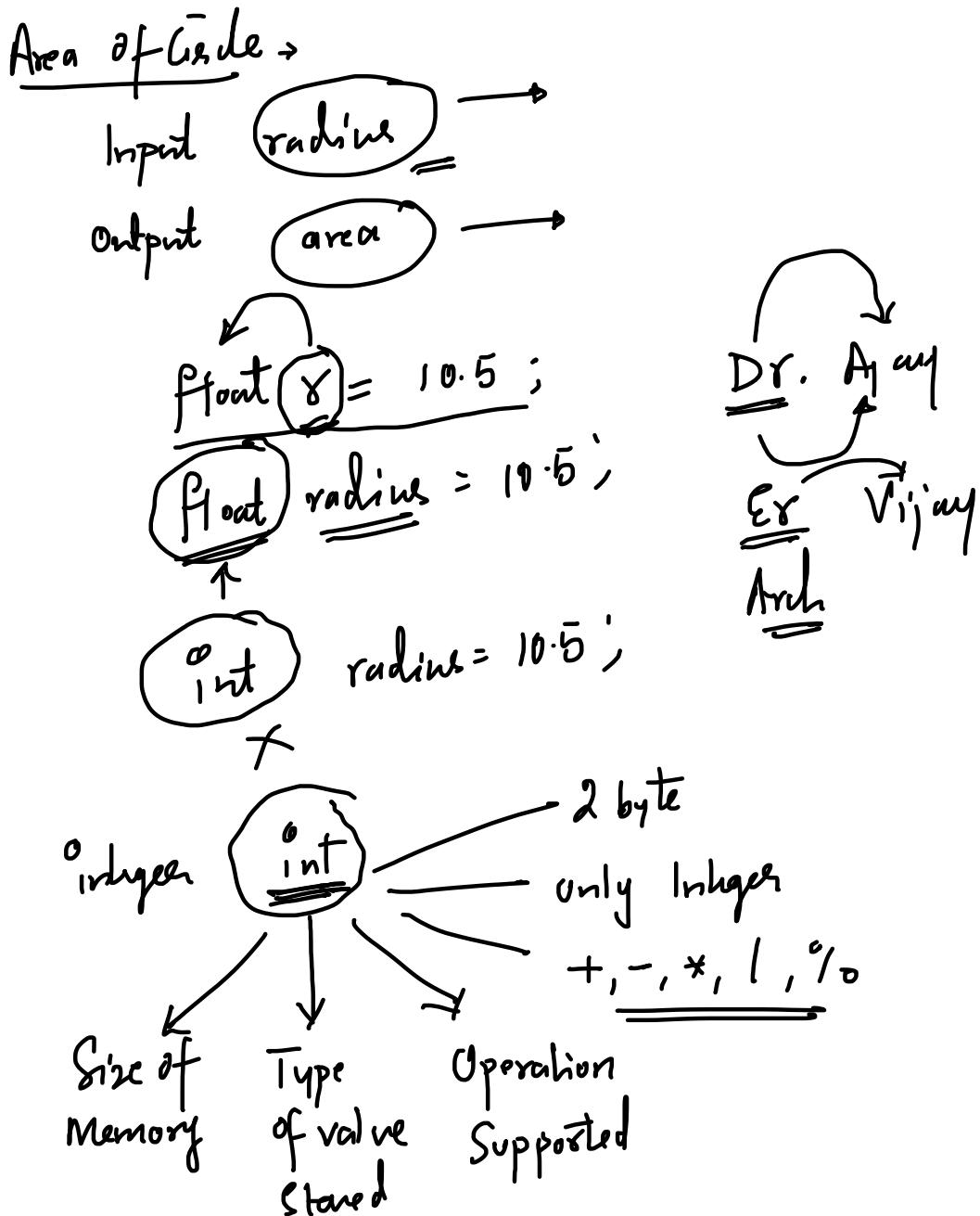
\* Requirement of memory depends on type of value we are going to deal with.

datatype

- ① Type of value you are going to deal with
- ② Size of Memory required to store it

- "
- ② Size of Memory required to store it
  - ③ Types of Operations that can be performed





## Data Types (Imp)

① It specifies the type of values that will be assigned to the memory

Basic Data Type

- ① Integer → int
- ② Floating point → float

## Built-In Type

- (2) floating point → float
- (3) character → char.

Void

⇒ Absence of value (no value is Available)

Derived Types ⇒ These are the Data types created

Using Built In Types →

Ex pointers, Array, function, Structure

## Built In Data Types →

### ① Integer:

(Signed) char → size ⇒ 1 byte

(Always in Single Quote)

Note

1 byte = 8 bit ⇒  $2^8$  values

128

-ve

256 values

-128 to -1 } use

Non -ve 128

0 to 127 } char set

Ex       $\begin{matrix} 00 \\ 01 \\ 10 \\ 11 \end{matrix}$  } with 2 bits       $2^2$  values

$\begin{matrix} 0 \\ 1 \end{matrix}$  } with 1 bit       $2^1$  value

$\begin{matrix} 000 \\ 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \end{matrix}$  }  $2^3 = 8$  values

$2^n$  values

ASCII Character Set ( American Standard Code for Information

# ASCII character Set ( American Standard Code for Information Interchange )

C uses ASCII character set  $\Rightarrow$

ASCII character set supports 128 characters

\* Complete C program can be written using 128 characters.

\* Every character has ASCII value (0 - 127)

\* ASCII values for few characters are  $\rightarrow 0 \underline{\hspace{1cm}} 127$

Upper Case	(	65 A	66 B	67 C	- -	90 Z	)
+ 32						- 32	
Lower Case	(	a 97	b 98	c 99	- -	z 122	)
digit	(	48 0	49 1	50 2	.	57 9	)

Ex char ch='a';

i) printf("%c", ch-32);  
if A

④ char ch='a';  
printf("%c", ch);  
op > a  
a . . . 'a' .

Ques A

② `printf("%c", ch);`

Ques a

③ `printf("%d", ch);`

Ques  $\Rightarrow 97 \} \text{ ASCII value of } \underline{\text{ch}}.$

④ `char ch = '9';`  
`printf("%d", ch);`

Ques 57

### Note

Integer Type

(only Integer type values supports Unsigned)

Signed Represent^n  $\Rightarrow$  Will support both -ve values & non -ve values

Unsigned Represent^n  $\Rightarrow$  Not support -ve values & will only support Non -ve values

### Significance

Consider

1 byte  $\Rightarrow 2^8$  values  
 $= 256$  values

Signed:  
-ve  $128$  -ve  $\Rightarrow -1$  to  $-128$   
non -ve  $128$  Non -ve  $\Rightarrow 0$  to 127

Unsigned:  $256$  Non -ve  $\Rightarrow 0$  to 255

Ex  $\rightarrow$   
To Assign Roll NO  $\Rightarrow$  Signed  
11 + 1 r 1

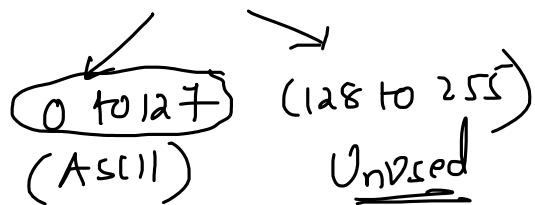
$\rightarrow$  Highest roll no  $\Rightarrow$  127



Note

When we are sure that we will deal with only Non-negative values we may use unsigned

unsigned char  $\Rightarrow$  1 byte  $\Rightarrow$  0-255



short (Signed)  $\Rightarrow$  2 bytes  $\Rightarrow$   $\begin{array}{l} -32768 \text{ values} \\ 65536 \text{ values} \end{array}$   $\Rightarrow$   $\begin{array}{l} -1 \text{ to } -32768 \\ 0 \text{ to } 32767 \end{array}$

Range  $\Rightarrow$   $\underline{-32768 \text{ to } 32767}$

unsigned short  $\Rightarrow$  2 bytes  $\Rightarrow$   $\underline{0 \text{ to } 65535}$

int (Signed)  $\Rightarrow$  2 bytes / 4 bytes  $\Rightarrow$  for 2 bytes  $\Rightarrow$   $\underline{-32768 \text{ to } 32767}$

16 bit System      32/64 bit System

Consider

System  
Consider

System

unsigned int  $\Rightarrow$  2 byte / 4 byte  $\Rightarrow$  For 2 byte  $\Rightarrow$  0 to 65535

System Architecture.

long  $\rightarrow$  4 byte  $\Rightarrow$   $2^{32}$  values

-ve { -2147483648  
to 2147483647  
Non-ve }

unsigned long  $\Rightarrow$  4 byte  $\Rightarrow$   $2^{32}$  values } only Non-ve } 0 to 4294967  
295

### Summary on Integer Data Type

char (signed)	1 byte
unsigned char	
short (signed)	2 byte
unsigned short	
int (signed)	2 byte / 4 byte
unsigned int	
long (signed)	4 byte
unsigned long	

### Floating Point Data Type \*

Here precision is imp  $\Rightarrow$  No of values displayed after floating point

float  $\Rightarrow$  4 byte  $\Rightarrow$   $1.2 \times 10^{-38}$  to  $3.4 \times 10^{38}$

Precision  $\Rightarrow$  6 decimal places

double  $\Rightarrow$  8 byte  $\Rightarrow$   $2.3 \times 10^{-308}$  to  $1.7 \times 10^{308}$

Precision  $\Rightarrow$  15 decimal places

long double  $\Rightarrow$  10 byte  $\Rightarrow$   $3.4 \times 10^{-4932}$  to  $1.1 \times 10^{4932}$

Precision  $\Rightarrow$  19 decimal places

## \* Format Specifiers

character  $\Rightarrow \%$  C

Signed integer  $\Rightarrow \%$  d,  $\%$  i

Unsigned integer  $\Rightarrow \%$  u

Floating point  $\Rightarrow \%$  f,  $\%$  e,  $\%$  E

Scientific notation of float

long  $\Rightarrow \%$  ld,  $\%$  l,  $\%$  li

Unsigned long  $\Rightarrow \%$  lu

short  $\Rightarrow \%$  hi

(Signed)

unsigned short  $\Rightarrow$  %hu.

double  $\Rightarrow$  %lf

long double  $\Rightarrow$  %Lf

%g or %G  $\Rightarrow$  floating point representation  
upto specified No of pos<sup>n</sup> after floating  
or max = 4

%s  $\Rightarrow$  for string (collection of character)

%o  $\Rightarrow$  octal Number System -

%X/%x  $\Rightarrow$  Hexadecimal System

## Number System

- ① Decimal Number System  $\Rightarrow$  Use 10 digits 0-9 to represent any value  
 \* Use in day to day life

- ② Binary No System  $\Rightarrow$  Use 2 digits 0 & 1 to represent any value

- ① Decimal to Binary Conversion  $\Rightarrow$  (for the value)

$$10 \Rightarrow \begin{array}{r} 2 | 10 \\ \hline 2 | 5 \quad 0 \\ \hline 2 | 2 \quad 1 \\ \hline 1 \quad 0 \\ \hline \end{array} \quad \underline{\underline{1010}}$$

- ② Binary to Decimal Conversion.

$$\begin{aligned} 1010 &\Rightarrow 1 \quad 0 \quad 1 \quad 0 \\ &\Rightarrow 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \\ &\Rightarrow 8 + 2 = \underline{\underline{10}} \end{aligned}$$

$$\overline{181'ch} \Rightarrow \begin{array}{cccccccccc} 512 & 256 & 128 & 64 & 32 & 16 & 8 & 4 & 2 & 1 \end{array}$$

$$(200)_2 \Rightarrow \begin{array}{ccccccc} & 0 & 0 & 1 & 1 & 0 & 0 \\ & \hline & & & & & \end{array}$$

$$(11001)_{10} \Rightarrow \begin{array}{ccccccccc} & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ & \hline & & & & & & & \end{array}$$

$$16 + 8 + 1 = \underline{\underline{25}}$$

Octal Represent<sup>n</sup>  $\Rightarrow$  We use 8 digits 0-7 to represent any value-

$$(96)_8 \Rightarrow \begin{array}{c|cc|c} 8 & 9 & 6 & \\ \hline 8 & 1 & 2 & 0 \\ & 1 & 4 & \\ \hline & 1 & & \end{array} = \underline{\underline{140}}$$

$$(140)_{10} \Rightarrow \begin{array}{ccc} 8^2 & 8^1 & 8^0 \\ | & 4 & 0 \\ \Rightarrow 1 \times 8^2 + 4 \times 8^1 + 0 \times 8^0 & \Rightarrow 64 + 32 = \underline{\underline{96}} \end{array}$$

Hexadecimal System  $\Rightarrow$  Use 16 digits to represent any value -

0 — 9 & A B C D E F

↓      ↓      ↓      ↓      ↓      ↓  
 10    11    12    13    14    15

\* Used to represent Address of Memory

Decimal to Hexadecimal Conversion →

$$(139)_{16} \Rightarrow \begin{array}{r} 16 | 139 \\ \hline 8 | 11 \rightarrow \underline{\underline{B}} \\ \hline 8 \end{array} = \underline{\underline{8B}}$$

Hexadecimal to Decimal

$$(8B)_{10} \Rightarrow 8^{16^1} + B^{16^0} \Rightarrow 8 \times 16 + 11 \times 1 \\ \Rightarrow 128 + 11 \Rightarrow \underline{\underline{139}}$$

Identifier  $\Rightarrow$  It's name assigned to the Memory  
to access the Memory

Rules ->

- ① Use Alphabets a - z and A - Z
- ② Use digits 0 - 9
- ③ Special character - (underscore)
- ④ First character will be Alphabet / underscore  
(must not start with digit)

Ex int 0fbond; X

int -0fbond; ✓

int bond0f; ✓

⑤ Must Not use Keyword (Predefined Reserved word with Predefined meaning)

⑥ Ideally length not more than 31 characters

Variable Type Memory  $\Rightarrow$  Content of Memory  
 $\times \boxed{10 \atop 20}$   
Can change during execution

Memory  
Type



`int x=10;  
x=20;`

- \* X refers to a variable type memory
- \* X is variable

Can change during execution

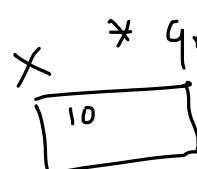
Constant Type Memory  $\Rightarrow$  Content of Memory

Cannot be changed

\* use keyword const

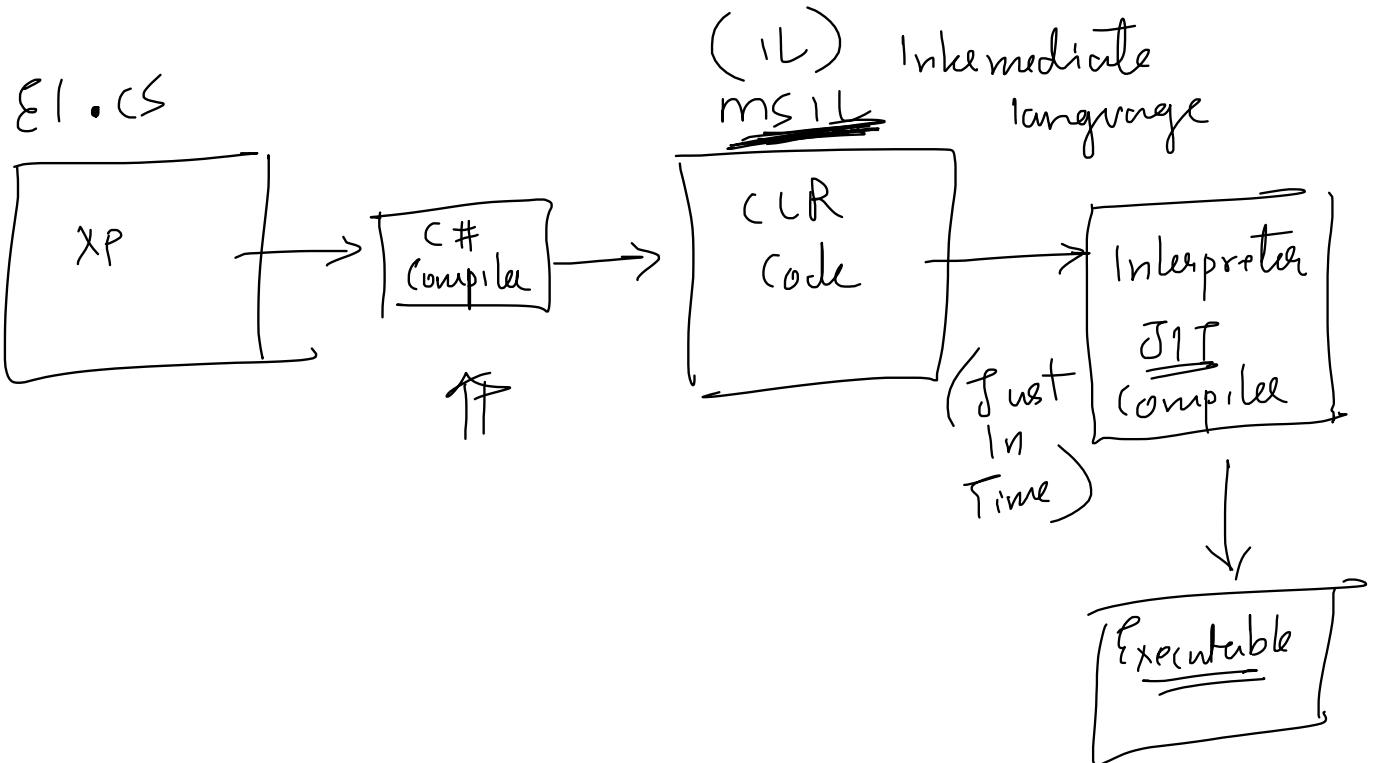
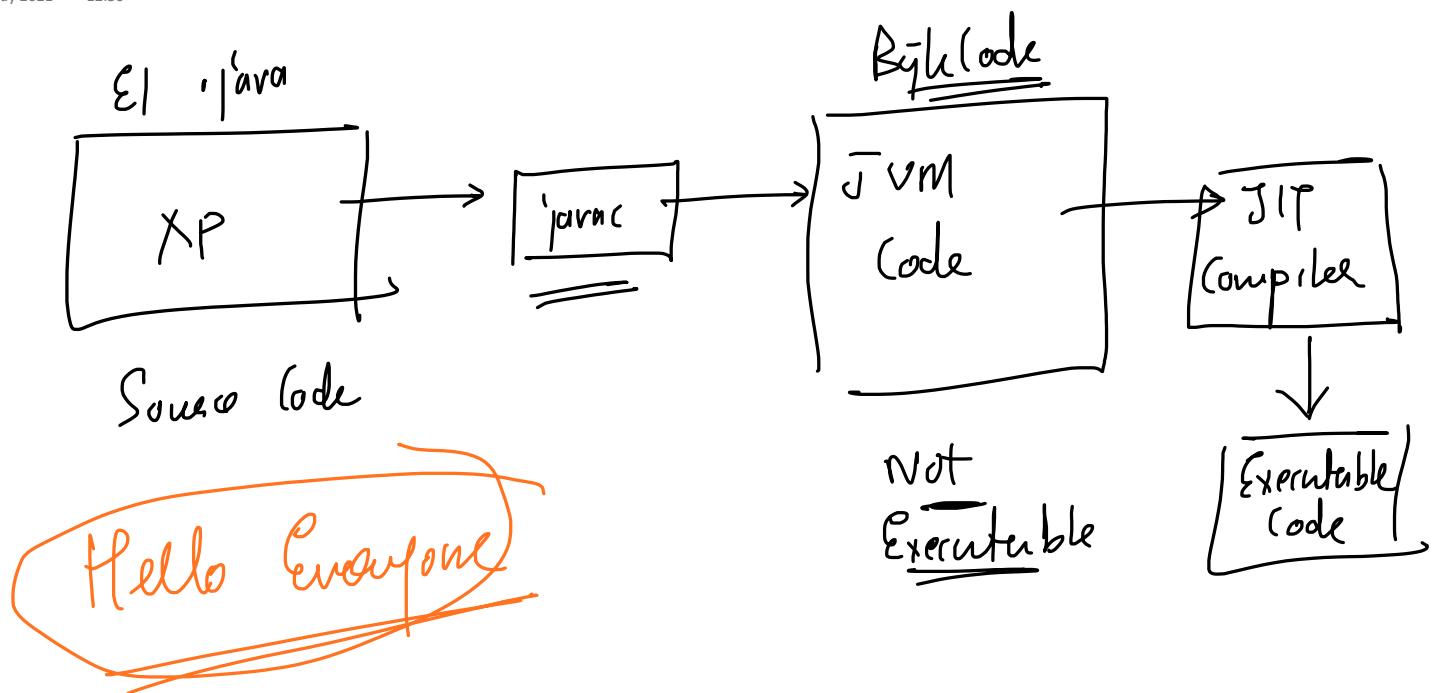
const `int x=10,  
x=20;`

X  
not allowed



\* Generate error message if we attempt to change -

\* X is constant



Hello Everyone

Hello everyone

EE

Ww

Hello Students

Hello Madam?

"Aao Study Karte Hain"

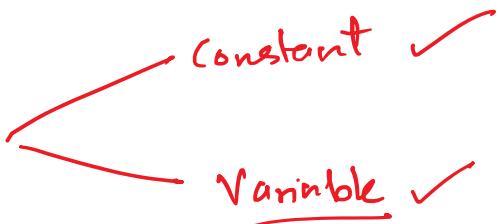
"Sub Moh Maaya Thai"

- ① Go to the border
- ② Click up

int  $\Rightarrow$  -32768 to 32767

Unsigned = 0 to 65535

Memory



~~const int x=10;~~  
(read only memory)

Keywords  $\Rightarrow$

- \* These are predefined words in C
- \* Having predefined meaning
- \* Meaning cannot be changed.

Eg

printf, scanf, int, float, char, long, long double, void, main

return -- - - - - -

## Operators in C.

### ① Arithmetic Operators.

- \* Binary Operators.
- \* 2 operands.

$+$   $\Rightarrow$  } All int & floating point  
 $-$   
 $*$   
 $/$   $\Rightarrow$  Returns Quotient of Division operation.

$$5/2 \Rightarrow 2$$

$$\underline{5.0}/\underline{2} \Rightarrow \underline{2.5}$$

$$2/5 \Rightarrow 0$$

$$\underline{2.0}/\underline{5} \Rightarrow \underline{0.4}$$

Even if one operand is floating point the division will be floating point

$2/\frac{5}{2} \Rightarrow$  If both the operands are Integer  
 then Integer division will take place.

$\%$   $\Rightarrow$  Returns Remainder of division operation.

[ Will work only with Integer values  
 Not work with floating point values ]

$$5/2 \Rightarrow 2 \frac{1}{2}$$

$5.4 \% 2 \times$  Not work (not supported)

## Relational Operators

- \* Operands are numeric values

{ Mon - 12:15 - 1:15  
 Wed: 11:15 - 1:15  
 Thurs: 9-11

\* Result is true / false (boolean value)

In C boolean type is not supported

[ In C      Non zero  $\Rightarrow$  true  
              Zero       $\Rightarrow$  false ]

true  $\Rightarrow$  1 / Non zero

false  $\Rightarrow$  0

$\underline{d0} > \underline{10} \Rightarrow \underline{\text{true}} \Rightarrow \underline{1}$   
↑  
numeric      Result

<  $\Rightarrow$  less than

$\leq \Rightarrow$  less than Equal to ( $\neq$ )

>  $\Rightarrow$  greater than.

$\geq \Rightarrow$  greater than Equal

$= = \Rightarrow$  Equal to (to compare Equality)

$\neq \Rightarrow$  Not Equal to .

Ex      0 int a=10, b=10;

$a == b \Rightarrow \text{true} \Rightarrow 1$

$a \neq b \Rightarrow \text{false} \Rightarrow 0.$

logical Operator  $\Rightarrow$  Operands are boolean (true / false) (Zero / Non zero)

Result are also boolean. (0/1)  
(logical)

&&  $\Rightarrow$  logical AND.

1 && 1  $\Rightarrow$  1

} 1 is not  
numerical  
! to true, 1 (true)

$\&$   $\Rightarrow$  Logical And.

$$\begin{array}{c} \text{1/0} \\ (a > b) \& \& (a > c) \\ \hline \quad \quad \quad \quad \quad \end{array}$$

1	$\&$	1	$\Rightarrow$	1	} numeric 1 its logic 1 (true)
1	$\&$	0	$\Rightarrow$	0	
0	$\&$	1	$\Rightarrow$	0	
0	$\&$	0	$\Rightarrow$	0	

1  $\Rightarrow$  True  
0  $\Rightarrow$  False.

! Logical NOT

$$!(\text{Zero}) = 1$$

$$!(5) = 0$$

$$!(\text{NonZero}) = 0$$

$$!(0) = 1$$

IMP

Bitwise Operator: \* Operands are binary form.

\* Result is Number in decimal form.

① Bitwise And Operator ( $\&$ )

1	$\&$	1	$\Rightarrow$	1
1	$\&$	0	$\Rightarrow$	0
0	$\&$	1	$\Rightarrow$	0
0	$\&$	0	$\Rightarrow$	0

} Bitwise  
Binary 1 (and)  
Binary 0.

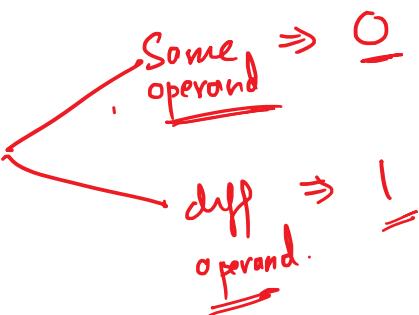
Eg  
 $\text{int } a = 10;$   
 $\text{int } b = 20;$   
 $\text{int } c = a \& b;$   
 $\Rightarrow 0$

$$\begin{array}{l} a = 128 \ 64 \ 32 \ 16 \ 8 \ 4 \ 2 \ 1 \\ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \\ \& b = 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \\ \hline \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \end{array}$$

② Bitwise OR Operator (|)

1		1	$\Rightarrow$	1
0		1	$\Rightarrow$	1
1		0	$\Rightarrow$	1
0		0	$\Rightarrow$	0

$$\begin{array}{r}
 1 \ 1 \ 0 \quad \Rightarrow \ 1 \\
 0 \ 1 \ 0 \quad \Rightarrow \ 0 \\
 \hline
 \text{Binary 1 and 0}
 \end{array}$$



### ③ Bitwise XOR Operator

Represent

$$1 \wedge 1 \Rightarrow 0$$

$$0 \wedge 0 \Rightarrow 0$$

$$0 \wedge 1 \Rightarrow 1$$

$$1 \wedge 0 \Rightarrow 1$$

### ④ n Bitwise Not (Returns 1's Complement)

$$\begin{aligned}
 \sim 1 &\Rightarrow 0 \\
 \sim 0 &\Rightarrow 1
 \end{aligned}
 \left. \right\}$$

$$\star (10)_a \Rightarrow \underline{\underline{0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0}}$$

$(-10)_2 \Rightarrow$  -ve decimal no ka binary kaise nikale.

Step 1  $\Rightarrow$  Take +ve version of the number  
10

Step 2  $\Rightarrow$  Convert  $\text{into binary form}$ .

$$0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0$$

Step 3  $\Rightarrow$  Find 2's complement of the binary form.

How to find 2's complement ① Find 1's complement (replace  $0 \rightarrow 1$  and  $1 \rightarrow 0$ )

② Add decimal 1 to it

(2) Add decimal 1 to it

$$\begin{array}{r} (10)_2 = 00001010 \\ \text{1's complement} \Rightarrow 11110101 \\ + 1 \quad \quad \quad + 00000001 \\ \hline 11110110 \end{array}$$

Binary Addition

Rule for binary addition

$$\begin{aligned} 1+0 &\Rightarrow 1 \\ 0+1 &\Rightarrow 1 \\ 0+0 &\Rightarrow 0 \\ 1+1 &\Rightarrow 0 \text{ carry 1} \end{aligned}$$

$$(-10)_2 \Rightarrow 11110110$$

$(-10)_2 \Rightarrow (10)_2$  ka 2'sc complement

Note > In Binary Represent<sup>n</sup>

Let say  
1 byte  $\Rightarrow$

10001111  
↑  
msb  
(most significant bit)  
↓  
LSB  
(least significant bit)

In Signed Represent

The msb does not represent any value, it represents sign of number

msb=1 the no is -ve

msb=0 the no is Non-ve

Dhyans-

01001101  
↑  
msb

Agar Binary Diya hai to Sabse Pehle  $\xrightarrow[\text{MSB}]{\text{chk}}$  1  $\Rightarrow$  -ve  
 0  $\Rightarrow$  Non -ve.

Represent " " Progra  


To Convert -ve Binary to Corresponding -ve Decimal

Consider Binary Given: 11110110 } Iska decimal kya hai?

ASK    Signed / Unsigned

Agal Unsigned  $\Rightarrow$  Normal Jindagi (a ke power se multiply karao)

$$2^7 \quad 2^6 \quad 2^5 \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0$$

⇒

Agar Signed -  $\Rightarrow$  Mentos Jindagi

$$\text{ruk karo msf} \Rightarrow \begin{smallmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ \downarrow & & & & & & \end{smallmatrix}$$

Chk Karo MSB  $\Rightarrow$    
 MSB is 1 So -ve Number

Agar msb is 1  $\Rightarrow$  Find 2's Complement of given binary.

$$\begin{array}{r}
 + 1 \quad + 0 0 \quad 0 0 \quad 0 \quad 0 \quad 0 \quad 1 \\
 \hline
 0 0 \quad 0 0 \quad 1 \quad 0 \quad 1 \quad 0
 \end{array}$$

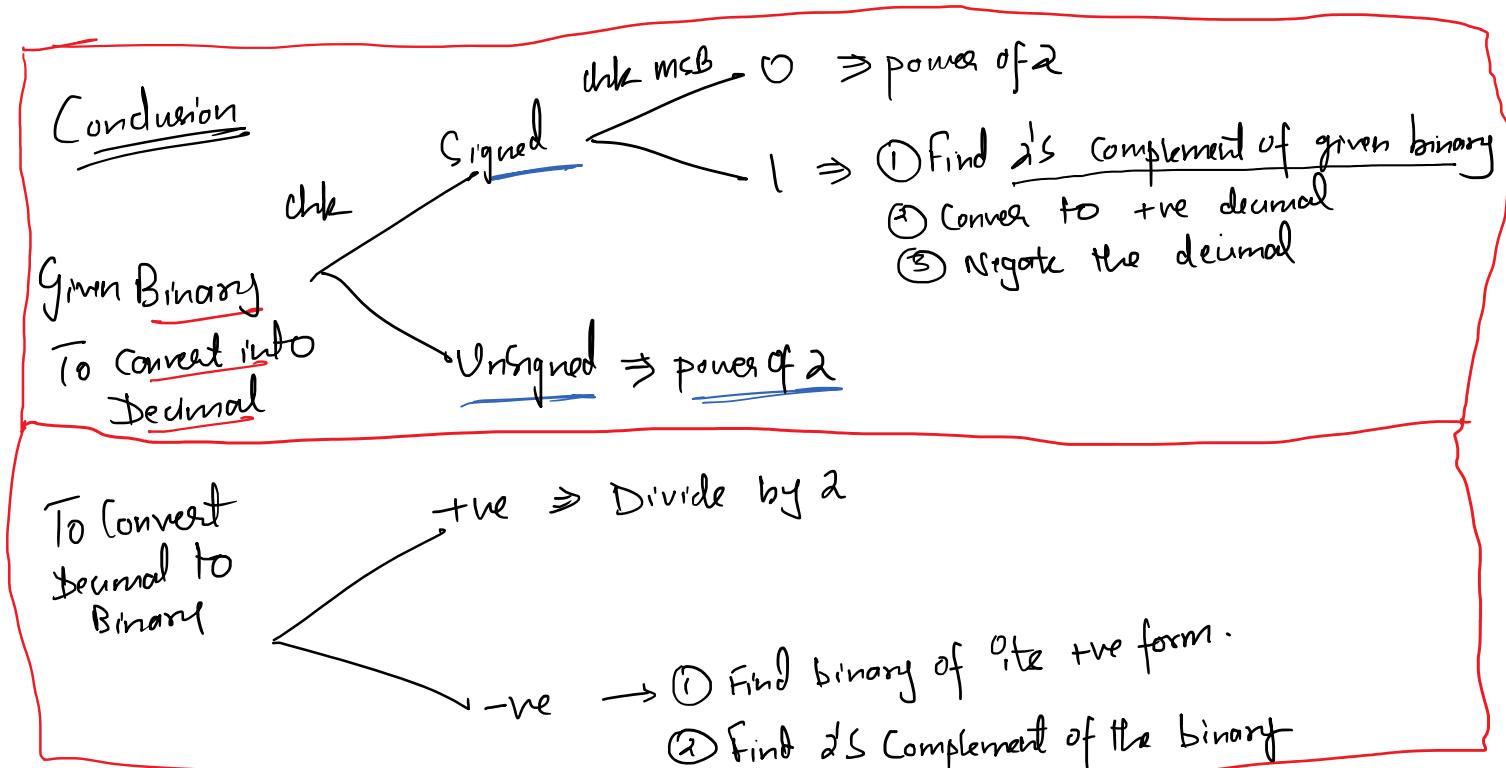
DSC of  
given

$\Rightarrow$  Find decimal of 2's Complement

Here = 0 0001010  $\Rightarrow$  10

$\Rightarrow$  Negate the value  $\Rightarrow \underline{\underline{-10}}$

Agar Signed ka msB O hair koh  $\Rightarrow$  Sadbhawan Jindagi  
(Power of 2)



Ex      Q math      2 byte  $\Rightarrow$  16 bit  
 $\text{int } a = 10;$   
 $\text{int } b = \underline{\underline{-a}};$   
 $= ?$   
 -11

$a = 00000000 \ 00001010 \Rightarrow$

$\downarrow a \Rightarrow 11111111 \ 11110101 \leftarrow$

msb=1

~~-11~~

$$\begin{array}{r}
 \text{d's} \Rightarrow \begin{array}{r} 00000000 \\ + 00000000 \\ \hline 00000000 \end{array} \quad \begin{array}{r} 00001010 \\ 00000001 \\ \hline 00000001 \end{array} \\
 \Rightarrow \underline{\underline{00000000}} \quad \underline{\underline{00001011}} \\
 = \boxed{11} \quad \underline{\underline{\text{Negate this}}} = \boxed{-11}
 \end{array}$$

## Bitwise left shift Operator ( $<<$ )

If shifts the bits to left by specified no of bit position.

\* The vacant places in right side gets filled with 0.

Ex

```

int a=10;
int b=a<<2;
            
```

$\Rightarrow 40$

$10 \times 2^2$   
 $= 10 \times 4 = 40$

$a << 2 \Rightarrow$

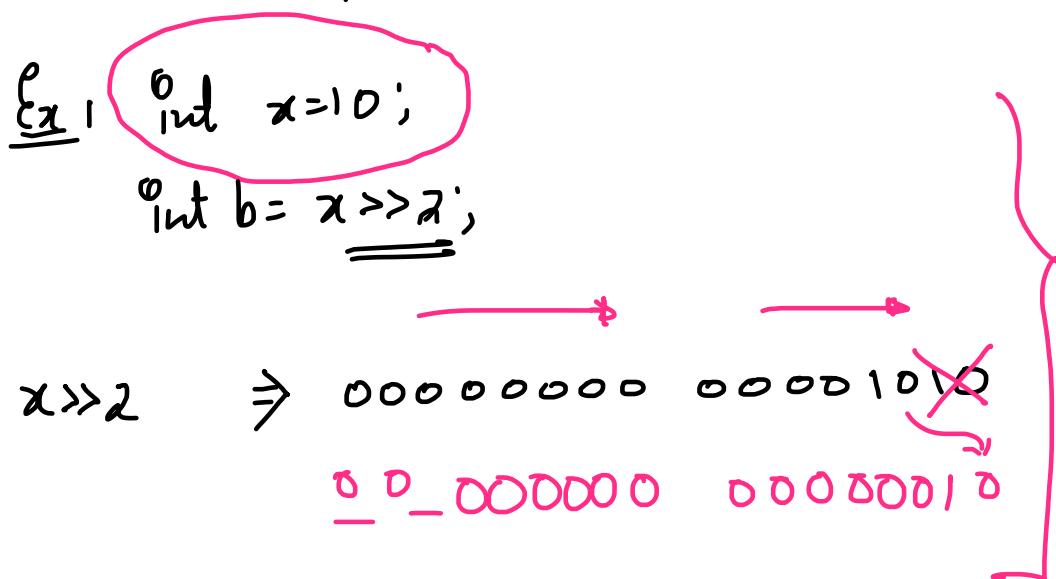
<del>00000000</del> <del>00000000</del>	4	<del>00001010</del> <del>00101000</del> <small>↓↓↓↓</small> <small>2 bits 4 r.s.</small>
--	---	---

$\Rightarrow 40$

Conclusion  $\Rightarrow$  Every time we left Shift a Number by  $K$  bits Result  $\Rightarrow$  Number  $\times 2^K$

## Right Shift Operator: ( $>>$ )

\* Here the bits are shifted to right by specified no of bit position



Ex 2  $\text{int } x = -10;$

$\text{int } b = \underline{\underline{x >> 2}};$

$$\begin{array}{r} 00000000 00001010 \\ 1's \downarrow 11111111 11110101 \\ + \\ \hline (-10)_2 \Rightarrow 11111111 11110110 \end{array}$$

$\Rightarrow 2^{\text{bit pos}^n}$

$$\begin{array}{l} \underline{\underline{x >> 2}} \Rightarrow 11111111 11110110 \\ \uparrow \quad \uparrow \\ \underline{\underline{11111111 11110110}} \end{array}$$

Note > When the bits are shifted to right there are blank positions in the left side (i.e msb end)

To fill the blank positions

If the number is Non -ve then it has to remain non -ve, So fill it with 0  
If the number is -ve then it has to remain -ve, So fill it with 1

Unary Operator  $\rightarrow$  The operator operates on Single Operand -

① Unary Minus (-)  $\rightarrow$  It simply changes the sign of decimal number

Ex       $\text{int } x=10,$   
 $\text{int } y = -\underline{\underline{x}},$   
$$\boxed{y = \underline{\underline{-10}}}$$

$\text{int } x=-10;$   
 $\text{int } y = -x,$   
 $= -(-10)$   
$$\boxed{y = 10}$$

② Increment Operator (++)

\* It increments the value of operand by 1

Normal Addition  
Consider  $\text{int } \underline{x} = 10,$

$x$   
 $\boxed{10}$

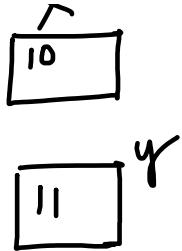
Increment Operation

$/$   
 $\underline{\underline{1}}$

Consider

$\text{int } \underline{x} = 10,$   
 $\text{int } y = \underline{x + 1};$

value of  
Memory  
identified  
by  $x$



However

$\text{int } \underline{x} = 10;$   
 $= \underline{x} = \underline{x + 1};$

represents  
address of  
Memory  
identified by  
 $x$



assignment operator

(Assigns value on RHS  
to address on LHS)

## Type 2 Post Increment Operator

Syntax  $\Rightarrow$  operand  $\underline{++}$

Working  $\rightarrow$  Step 1) Post Increment  $\underline{++}$  ko ignore karo

Type 1  $\Rightarrow$  Pre Increment Operator

Syntax  $\Rightarrow$   $\underline{++}$  operand

Working  $\Rightarrow$

① Get first increments the value of operand.

② Now ignore the preincrement operator

③ Solve the Exp.

Ex  $\text{int } \underline{x} = 10;$

$\text{int } y = \underline{\underline{++x}} + 5;$

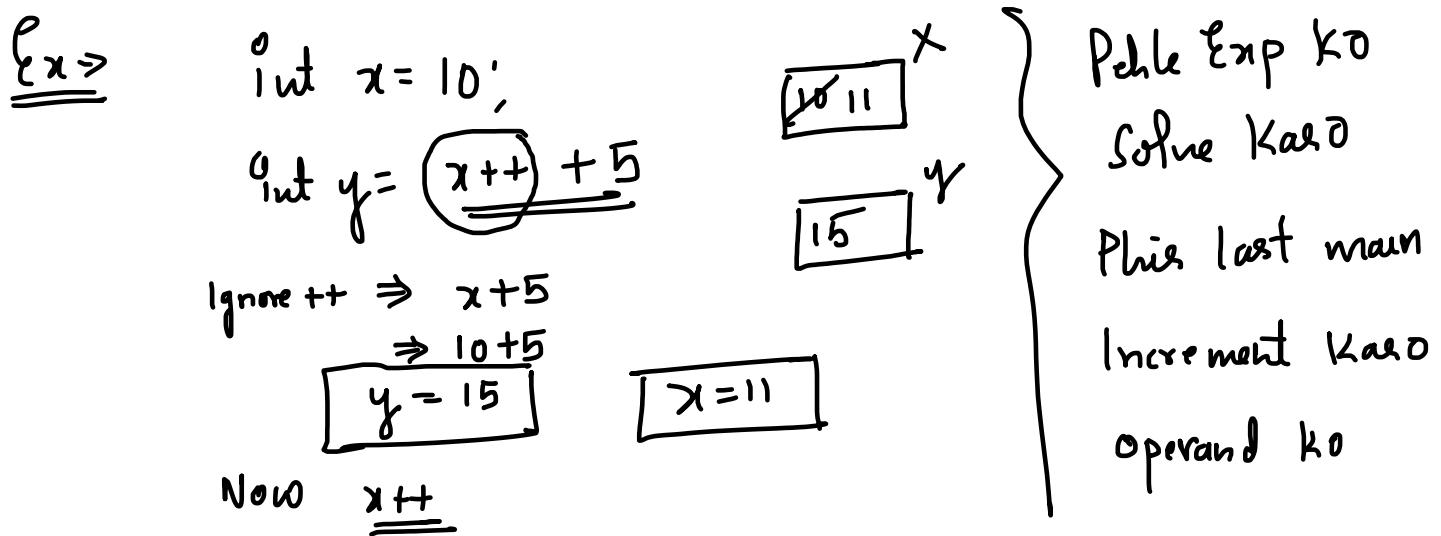


$$\Rightarrow y = \underline{x + 5}$$

\* (Preincrement Means

Pehle Increment Karo  
Phir Exp Main Use Karo)

- Working → Step 1) Post Increment ++ ko ignore karo  
 Step 2) Exp solve karo  
 Step 3) RHS ka value LHS ko Assign karo.  
 Step 4) Ab operand ko ++(Increment) karo



## Decrement Operator (- -)

\* It decrements the value of Operand by  $\frac{1}{=}$

## Pre Decrement Operator

Syntax     $-- \text{operand}$

### Working

- ① first perform decrement in operand
- ② Ab ignore --
- ③ Solve karo

int  $x = 10;$

$x$   
10 9

int  $y = --x + 5$

$$\begin{array}{l}
 \text{int } y = \underline{-x} + 5 \\
 \Rightarrow x + 5 \\
 \Rightarrow 9 + 5 \\
 \boxed{y \Rightarrow 14}
 \end{array}
 \quad
 \begin{array}{c}
 \boxed{7} \\
 | \\
 \boxed{14} \quad y
 \end{array}$$

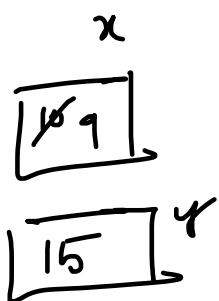
## Post Decrement Operator

Syntax > Operand --

- Working
- ① Initially -- ko  
ignore ko'd
  - ② Exp Solve ko'd
  - ③ Assign RNS  $\rightarrow$  LNS
  - ④ Now perform Decrement Operation

Ex

$$\begin{array}{l}
 \text{int } x = 10; \\
 \text{int } y = \underline{x--} + 5 \\
 \text{Ignore --} \Rightarrow x + 5 \\
 \text{Solve} \quad \boxed{y \Rightarrow 15}
 \end{array}$$



Decrement x--

## Example on Increment & Decrement

Trick #1 Pehle Soare Pre Operation Karo (Increment/Decrement)

(2) Ab Pre & Post Operations Ignore Karo

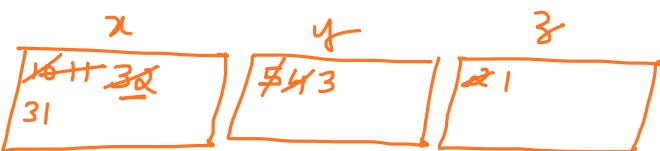
(3) Exp Ko Solve Karo

(4) Now Assign Ans  $\rightarrow$  LHS

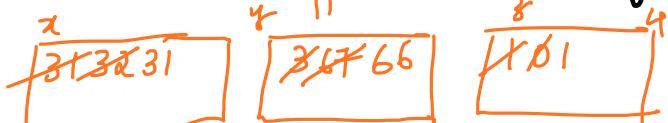
(5) Now perform All Post Operation (Increment/Decrement)

Consider

int  $x=10, y=5, z=2$



$$x = \underline{x--} + \underline{--y} + \underline{++x} + \underline{y--} + \underline{z--},$$



$$\Rightarrow y = \underline{--z} + \underline{y--} + \underline{2++} + \underline{x--} + \underline{++x}.$$



$$z = \underline{z--} + \underline{y++} + \underline{x--} + \underline{--x} - \underline{--y};$$

$$\left. \begin{array}{l} x = ? \\ y = ? \\ z = ? \end{array} \right\} \quad \begin{array}{l} 29 \\ 66 \\ 60 \end{array}$$

z = ?    Go    |

Consider    int ~~x-2~~  $a=5$ ,  $b=7$ ,  $c=-3$

$$a = b++ + --c - a--$$
$$\underline{7} + (-3) - c-$$

$$a = -b + \cancel{c++} + --b + a-- + ++c$$
$$\underline{\underline{6}} + (-2) + (6) + (2) + (-2)$$
$$a = -b - c- + ++b$$
$$-4 - 7 + (-1) + (7)$$

$$a = ? , b = ? , c = ?$$
$$-4 \quad 6 \quad -6$$
$$\cancel{\cancel{\cancel{}}}$$

Assignment Operator  $\Rightarrow (=)$

General form      LValue = RValue

$\Rightarrow$  value specified by RValue is assigned to address specified by LValue

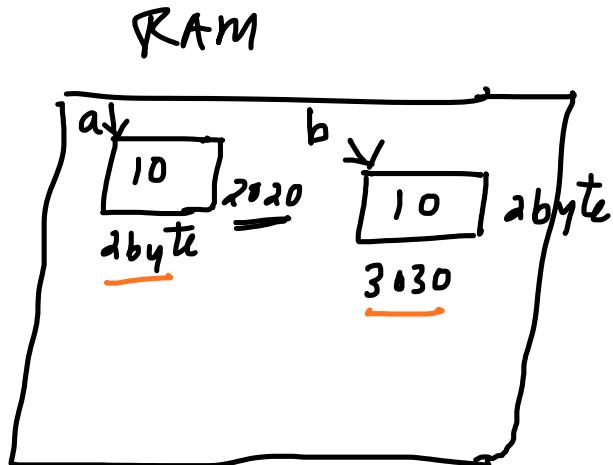
Ex

$\left\{ \begin{array}{l} \text{int } a = 10; \\ \text{int } b; \end{array} \right.$

$b = a;$   
 address of  $b$       value of  $a$

$3030 = 10$

Assign value 10 to the address 3030



Shorthand Assignment  $\Rightarrow$

$$+= \Rightarrow a += b; \Rightarrow a = a + b$$

$$-= \Rightarrow a -= b; \Rightarrow a = a - b$$

$$*= \Rightarrow a *= b; \Rightarrow a = a * b$$

$$/= \Rightarrow a /= b; \Rightarrow a = a / b$$

$$\% = \Rightarrow a \% = b; \Rightarrow a = a \% b$$

$$<< = \Rightarrow a <<= b, \Rightarrow a = \underline{\underline{a << b}}$$

Ex     $c <<= 2 \Rightarrow c = \underline{\underline{c << 2}}$

$$>> = \Rightarrow a >>= b; \Rightarrow a = a >>b;$$

Ex     $c >>= 2; \Rightarrow c = \underline{\underline{c >> 2}};$

$$\& = \Rightarrow a \&= b; \Rightarrow a = a \& b;$$

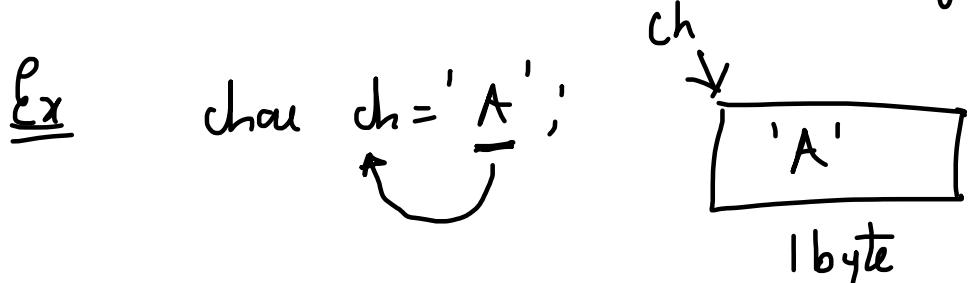
(bitwise  
And)

$$| = \Rightarrow a | = b; \Rightarrow a = a | b;$$

$$\wedge = \Rightarrow a \wedge = b; \Rightarrow a = a \wedge b.$$

## Character Constants $\Rightarrow$

Character literals are enclosed in single quotes



`printf("%c", ch);  $\Rightarrow$  A`

`printf("%d", ch);  $\Rightarrow$  65`

## Escape Sequences $\Rightarrow$

\n  $\Rightarrow$  New Line  $\Rightarrow$  It takes the cursor to new line

\a  $\Rightarrow$  Alert or bell

\b = Backspace [It takes cursor back by one position]

Ex    `printf("hello\bKaiseho");`

Output    hellKaiseho

\r = Carriage Return [Takes cursor to the start of the line]

Ex    `printf("hello\b\rKaise");`

- `printf(" hello bhai \x & Kaue ),`

\t      hello bhai-\t  
          Raise  
          Hello bhai  
 $\Rightarrow$  Kause bhai

\t       $\Rightarrow$  horizontal tab

\v       $\Rightarrow$  Vertical tab

\|       $\Rightarrow$  print \

\'       $\Rightarrow$  print ' (single quote)

\"       $\Rightarrow$  print " (Double quote)

\?       $\Rightarrow$  print ? (print question mark)

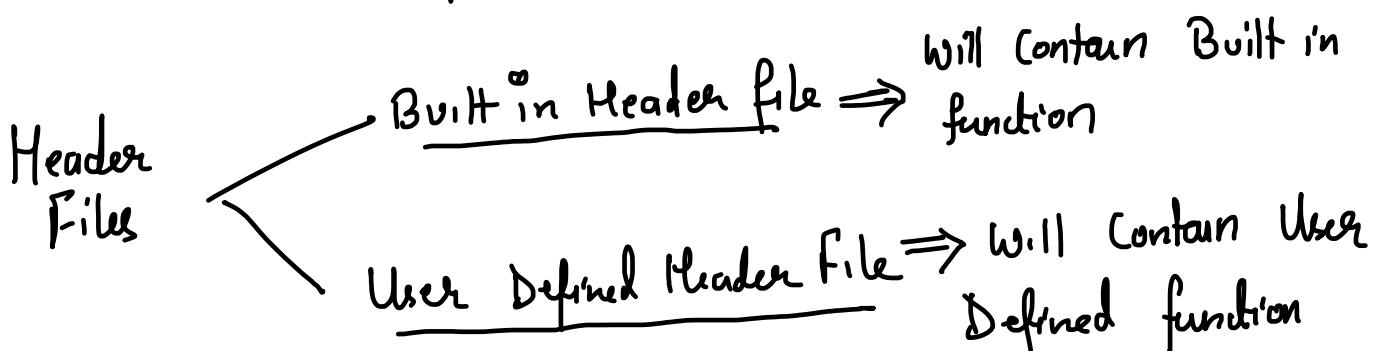
## Library Functions

Library → In C Library is collection of functions

performing operation on similar type

→ This library is known as Header file

For Ex ⇒ To perform input ⇒ scanf() } } <stdio.h>  
To perform output ⇒ printf()



## Built In Header file / library

1) <math.h> Header file → It is library that contains functions that performs Mathematical Operation

→ We must include <math.h> header file -  
functions are → Accepts float as input & returns float

① sqrt(x) ⇒ return square root value of x

② pow(x,y) ⇒ return  $x^y$

- ③  $\exp(x) \Rightarrow$  return  $e^x$
- ④  $\text{fabs}(x) \Rightarrow$  return absolute value of  $x$
- ⑤  $\log(x) \Rightarrow$  return natural log of  $x$   
(base e)
- ⑥  $\log_{10}(x) \Rightarrow$  return log of  $x$  (base 10)

Note  
⑧  $\%$  → does not work with floating point values  
Will only work with Integer values

If we want remainder value of floating point division (we cannot use %.)

In Such Case We Use

fmod(x, y) ⇒ Return remainder of floating point division of values  $x \underline{/} y$

⑨  $\sin(x)$  ⇒ Returns Sine of angle  $x$   
( $x$  must be in radian)

⑩  $\cos(x)$  ⇒ Returns Cosine of angle  $x$   
( $x$  must be in radian)

⑪  $\tan(x)$  ⇒ Returns tangent of angle  $x$   
( $x$  must be in radian)

for Arc Angles \*  $\text{asin}(x)$ ,  $\text{acos}(x)$ ,  $\text{atan}(x)$

(12)  $\text{floor}(x)$   $\Rightarrow$  Ex float  $x = 3.14$ ;  
Int values  $< x \Rightarrow 3, 2, 1, 0, -1, -2, \dots$   
largest = 3

$$\text{floor}(3.14) \Rightarrow 3$$

returns largest integer less than or equal to  $x$

(13)  $\text{ceil}(x)$   $\Rightarrow$  Ex float  $x = 3.14$ ;  
Int value  $> x \Rightarrow 4, 5, 6, 7, \dots$   
smallest  $\geq 4$   
 $\text{ceil}(3.14) = 4$

\* It returns smallest integer greater than or equal to  $x$

II) <ctype.h>  $\Rightarrow$  contains functions that are useful for testing & Mapping Characters

\* Those functions will take int as input (that) and will return int value

and will return int value

① int isalpha(int ch)  $\Rightarrow$  returns 1 if ch is alphabet  
else returns 0

② int isdigit(int ch)  $\rightarrow$  return 1 if ch is digit  
else return 0

③ int isalnum(int ch)  $\rightarrow$  return 1 if ch is alphanumeric  
(Alphabet or Number)

④ int islower(int ch)  $\rightarrow$  return 1 if ch is lowercase  
else return 0

⑤ int isupper(int ch)  $\rightarrow$  return 1 if ch is uppercase  
else return 0

⑥ int toupper(int ch)  $\rightarrow$  returns uppercase version  
of character ch.

⑦ int tolower(int ch)  $\Rightarrow$  returns lowercase version  
of character ch

⑧ int isspace(int ch)  $\Rightarrow$  return 1 if character ch  
is space  
else return 0

# Operator Precedence in C

Category	Operator	Associativity
Postfix	( ) [] -> . ++ --	Left to right
Unary	+ - ! ~ <del>++ --</del> (type)* & sizeof	Right to left →
Multiplicative	* / %	Left to right
Additive	+ -	Left to right →
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR		Left to right
Logical AND	&&	Left to right
Logical OR		Left to right
Conditional	? :	Right to left →
Assignment	= += -= *= /= %= >>= <<= &= ^=  =	Right to left →
Comma	,	Left to right

Priority

Associativity → If in Expression we have More than one operator of Same precedence than the order of Execution is decided by Associativity

Ex 1) 
$$\begin{array}{r} 3 + 4 * 5 \\ \hline 3 + 20 \end{array}$$

Here 2 operators of different precedence

$\Rightarrow \underline{\underline{2}} \underline{\underline{3}}$

Ex 2) 
$$\begin{array}{r} \overrightarrow{3 + 5} - 4 \\ \hline 8 - 4 \end{array}$$

$\Rightarrow$  2 operators of same precedence

$$8 - 4 \\ = \underline{\underline{4}}$$

Ex 4

$$\underline{\underline{1}} = \underline{\underline{2}} ! = 3$$

$$0 \underline{\underline{!}} = 3$$

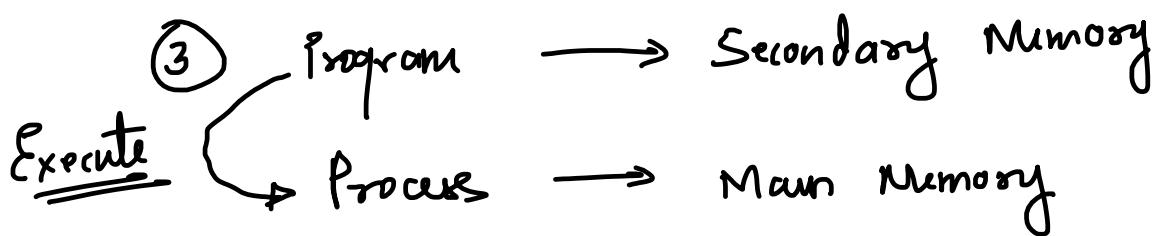
$$\Rightarrow \underline{\underline{1}}$$

Both Operator same  
precedence

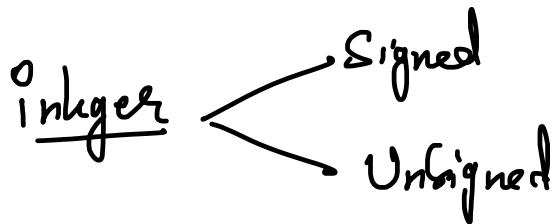
# You Must Know by Now?

① low level language & High level language

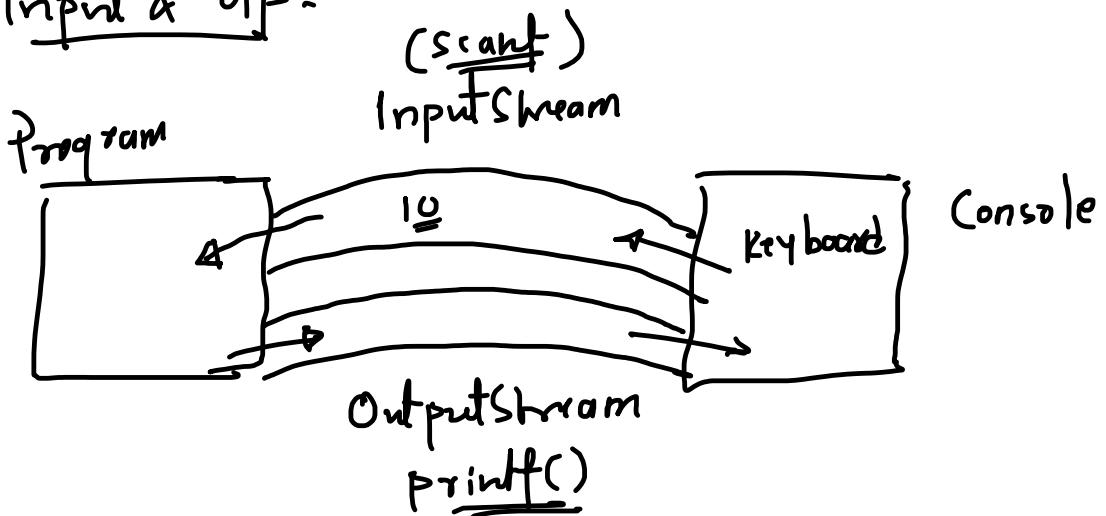
② Compiler & why need it



④ Data Type      Built in Types | Size |

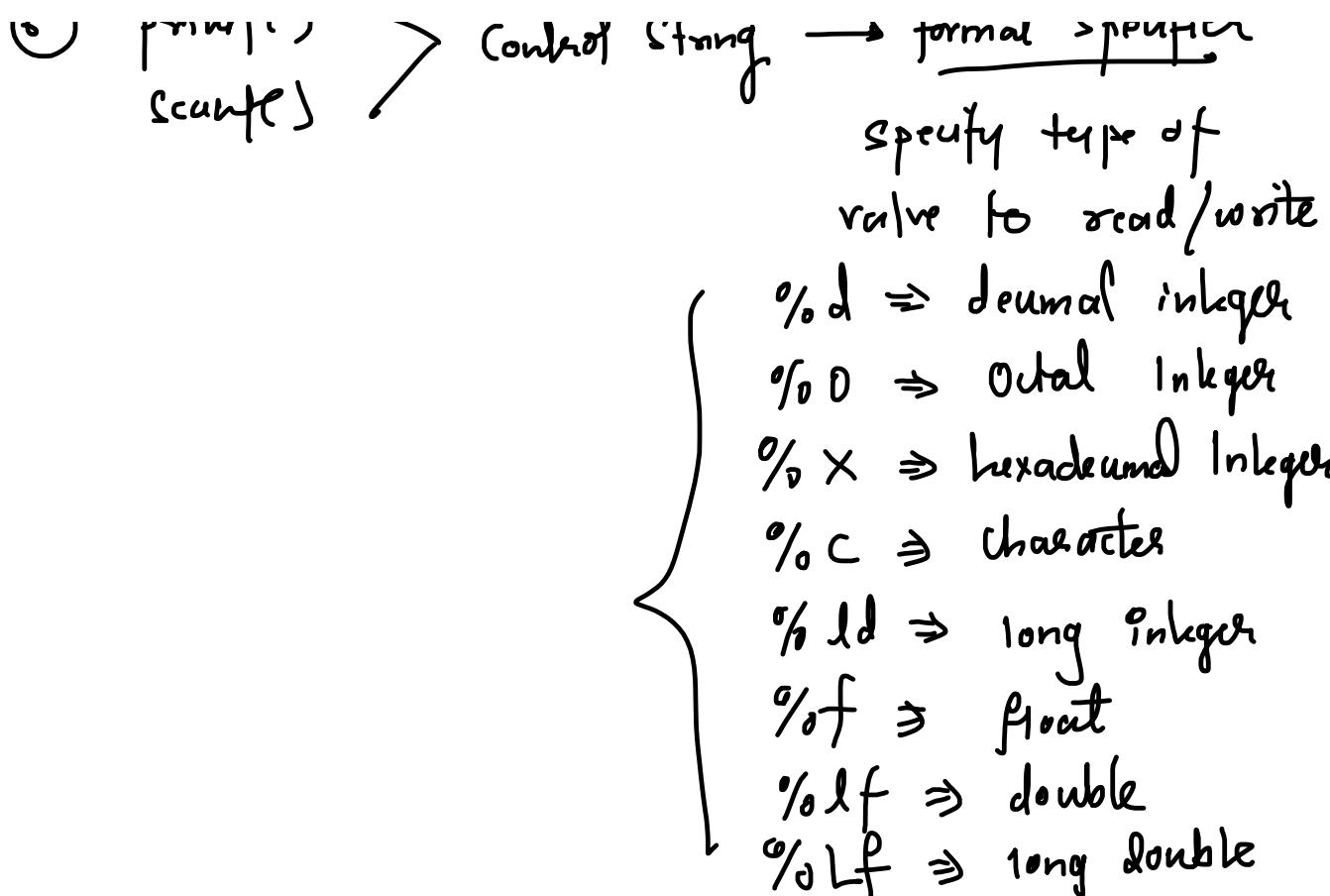


⑤ Input & Output



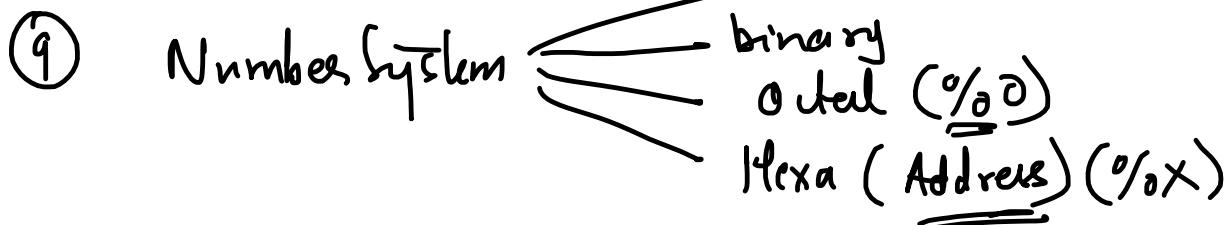
printf() & scanf() → <stdio.h>

⑥ printf()  
scanf() > Control String → format Specifier



- ⑦ Operators → Arithmetic  
 Boolean  
 Relational  
 Bitwise  
 Unary

- ⑧ Libraries ① <math.h>  
 ② <ctype.h>



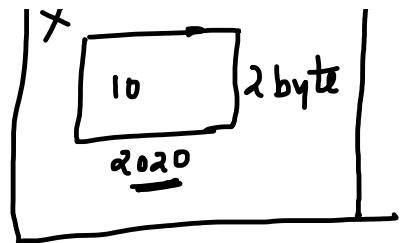
Note  
`int x; }`



Note

```

int x = 10;
scanf("%d", &x);
    
```



```

{ printf("%d", x);      => 10
  printf("%x", x);      => 2020
    
```

## Type Conversion <sup>①</sup> in C →

It is mechanism of converting value of one type to another

Type conversion is of Two Types: →

### ① Implicit Type Conversion - (Automatic)

- \* It is done by compiler on its own without external trigger by developer.
- \* When value of lower data type is used in Exp with higher data type then the lower data type is implicitly converted to higher data type

$$\text{Ex: } \text{float } y = 9 / 2 \Rightarrow 4.000000$$

↓      ↑  
int    int

$$\text{float } y = 9.0 / 2 \Rightarrow 9.0 / 2.0 \Rightarrow 4.500000$$

↓      ↓  
float    float

\* There <sup>is</sup> no loss of precision.

## Explicit Type Conversion / Type Casting →

Consider

$\text{float } x = 24 \underline{376852};$   
 $\text{int } y = (\text{int}) \underline{x}, \Rightarrow 24$

$\begin{matrix} \text{int} & \neq & \text{int} \\ \uparrow & \text{Casting} & \downarrow \\ \text{float} & & \end{matrix}$

Explicit

When value of higher type is to be assigned to memory of lower type, then we need to explicitly specify the Cast

i.e. We need to use Cast operator to explicitly convert value of higher type to lower type

\* There is loss of precision.

## Misc Operators

- ① sizeof() → Returns the size of argument passed in terms of byte

Ex

float x = 3.14;

printf("%d", sizeof(x)); ⇒ 4

printf("%d", sizeof(float)); ⇒ 4.

- ② & → Returns address of the identifier

[ Is memory ka naam diya jala address return karega ]

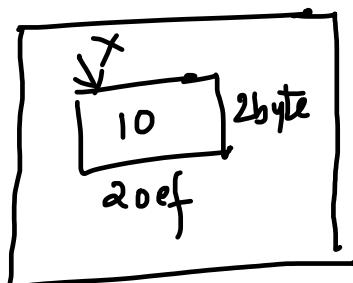
⇒ Type of Memory address: → hexadecimal (%x)  
RAM

Ex

int x = 10,

printf("%x", &x);

⇒ 20ef



- ③ ?: → Conditional Operator  
→ Ternary Operator  
→ (Uses 3 operands)

General form →

General form  $\Rightarrow$

$$(\text{exp1}) ? (\underline{\text{exp2}}) : (\text{exp3}) ;$$

Working  $\Rightarrow$  exp1 is evaluated

- if True  $\Rightarrow$  return the exp2 (Answer)
- if false  $\Rightarrow$  return the exp3 (Answer)

$$\begin{aligned} &\text{int } x=10, y=6; \\ &\text{int } z; \\ &z = (\underline{x > y}) ? \underline{x} : \underline{y}; \\ &\quad \text{exp1} \quad \text{exp2} \quad \text{exp3} \\ &\quad \text{true} \end{aligned}$$

Z = 10

To assign lower value

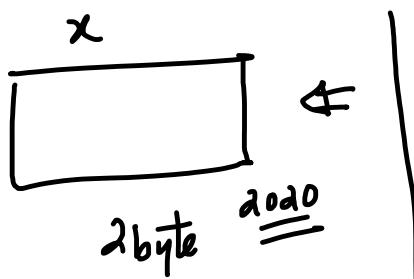
$$z = (\underline{x \leq y}) ? x : y;$$

false

Z = 6

Literal values  $\rightarrow$  value of the specified type

int  $\underline{x}$  ;  
 identifier

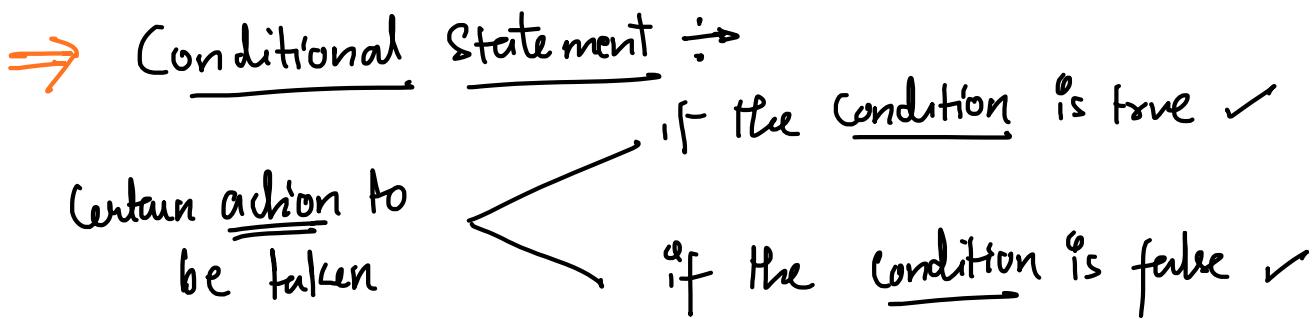


float  $\underline{x}$  ;  
 $x = 3.14$  ;  
 float  
 literal  
 value

$x = \underline{10}$   
 integer literal value

char  $\underline{x}$  ;

$x = 'A'$   $\leftarrow$  character literal value



## ① if

Syntax > if (condition)

if block  $\begin{cases} \rightarrow 1 & \text{Statement to be executed when condition is true} \\ \rightarrow ? & \end{cases}$

\* Agar condition true hua to kya karna ka

\* Agar condition false hua to kya karne ka ye nahin

bataenge

#include <stdio.h>

Ex int main()

{ int a,b;

printf("Enter values of a and b\n");

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

if( a > b)

{ printf("a is greater than b\n");

printf(" This is if example\n"),

}

return 0;

Case 1

a=10

b=20

if

—

Case 2

a=20

b=10

if

a is greater than b  
This is if example

Case 3

return 0;  
}

Case 3  
 $a=10$   
 $b=10$

if

Note if a single statement is supposed to be executed if the condition is true OR (if block contains only single statement) then the curly bracket can be ignored

Ex Case 1

```
if (a>b)  
{  
    printf("a is greater than b\n");  
    printf("Example of if\n");  
}
```

Case 2

```
if (a>b)  
{  
    printf("a is greater than b\n");  
    printf("Example of if\n");  
}
```

if  $a=10, b=10$

Case 1) if  $a > b \Rightarrow a$  is greater than b  
Example of if

Case 2) if  $a > b$   
Example of if

if  $a=10, b=20$

Case 1) if

Case 2) Example of if

\* Rule → if curly bracket is not specified then only one statement is associated with condition

One Statement is associated with condition  
(immediate next stat after condition)

## ② if - else

Here we specify action to be taken when Condition is true, we also specify action to be taken when condition is false

Syntax → if ( condition )

{      if block }      stmt to ^ executed when condition  
}                          is true

else  
{      else block }      stmts to ^ be executed when  
}                          condition is false

Ex → #include< stdio.h >

int main()

{      int a, b;

printf("Enter values of a and b \n");

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

if (a > b)

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

```

    1 printf("a is greater than b \n");
    }
else
    { printf("a is not greater than b \n");
    }
return 0,
}

```

- \* If more than one statement is desired to be executed when cond" is true(if block) or the condition is false(else block) then the curly bracket must be there
- \* If the if block or else block contains only single statement then the curly bracket can be avoided

### ③ else if ladder →

here we can specify different actions for different conditions

#### Syntax

```

if (condition 1)
{
    Condition 1 block
}
else if (condition 2)
{
    Condition 2 block
}
else if (condition 3)
{
    Condition 3 block
}
else
{
    else block
}

```

Here

- ① Condition 1 is evaluated
  - if true  
Condition 1 block Executed  
 all other blocks are bypassed
  - if Condition 1 is false  
 then Condition 2 is evaluated
  - if Condition 2 is true  
 then Condition 2 block is Executed  
 and all others are bypassed
  - if Condition 2 is also false  
 then Condition 3 is evaluated
  - if Cond^n 3 is true  
 then Cond^n 3 block is evaluated
  - if Cond^n 3 is also false

(ie All the cond' are false)  
Then else (last) block  
is Executed

Ex

```
#include <stdio.h>
int main()
{
    int a, b;
    printf("Enter value of a and b\n");
    scanf("%d %d", &a, &b);
    if (a > b)
        printf("a is greater than b\n");
    else if (a < b)
        printf("a is less than b\n");
    else
        printf("a and b are equal\n");
    return 0;
}
```

Q) Prog in C to read year no and determine  
year is leap year or not.

Logic  $\Rightarrow$  Let  $y$  be year no (Every  $4^{\text{th}}$  year is leap year)

If  $(\underline{\underline{y \% 400 == 0}})$  then year is leap year

Else

if  $(\underline{\underline{(y \% 4 == 0)}} \& \underline{\underline{(y \% 100 != 0)}})$

$\begin{matrix} \text{year is} \\ \text{divisible by} \\ 4 \end{matrix} \qquad \begin{matrix} \text{year is not} \\ \text{divisible by} \\ 100 \end{matrix}$

Then it is leap year

Else

year is not leap year

$$\text{Ex} \quad y = \underline{\underline{2021}}$$

$$\text{as } (\underline{\underline{y \% 400}}) \rightarrow 2021 \% 400 \Rightarrow \neq 0$$

then check  $F$   $\underline{\underline{(y \% 4 == 0)}}$   $\&\&$   $\underline{\underline{(y \% 100 != 0)}}$   $\Rightarrow$  False

False  $2021 \% 4 \neq 0$   $\&\&$   $2021 \% 100 \neq 0$  True

False  $2021 \% 4 \neq 0$  so answer is 0  
=====

then  $y$  is not leap year

①  $y = 1900$

check if  $y \% 400 == 0$  i.e.  $1900 \% 400 \neq 0 \Rightarrow$  False

then check  $\frac{True}{(y \% 4 == 0)}$  &  $\frac{False}{(y \% 100 != 0)} \Rightarrow$  False  
 $1900 \% 4 == 0$   $1900 \% 100 != 0$

∴ 1900 is not a leap year

②  $y = 2020$

check  $y \% 400 == 0$  False

Check if  $(y \% 4 == 0)$  &  $(y \% 100 != 0) \Rightarrow$  True

$2020 \% 4 == 0$

Yes  
(True)

$2020 \% 100 != 0$

(True)

∴ 2020 is leap year

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

Case 1) \* Every 400<sup>th</sup> year is leap year  
else  
removes every 4<sup>th</sup> year but not

```
int main()
{int y;
printf("enter the year\n");
scanf("%d",&y);
if( (y%400==0) || ( y%4==0 && y%100!=0 ))
printf("%d is a leap year\n",y);
else
printf("%d is not a leap year\n",y);
return 0;
}
```

checks every 4<sup>th</sup> year but not  
100<sup>th</sup> year is leap year

else not a leap year

⑧ To read month no from User and display  
no of days in the month

Logic  $\rightarrow$  Let 'm' be the month number

if ( m == 1 ) || m == 3 ) || m == 5 ) || m == 7 ) || m == 8 ) || m == 10 ) ||  
m == 12 )

✓ printf(" 31 days \n");

else if ( m == 4 || m == 6 || m == 9 || m == 11 )

- printf(" 30 days\n");

else if ( $m = 2$ )

```
d     print("Enter the year (\n");
```

```
Scanf("%d", &y);
```

`;f((y%400==0)||(y%4==0 && yy1w!=0))`

```
printf("29 days\n");
```

else

2      use    printf("28 days\n");

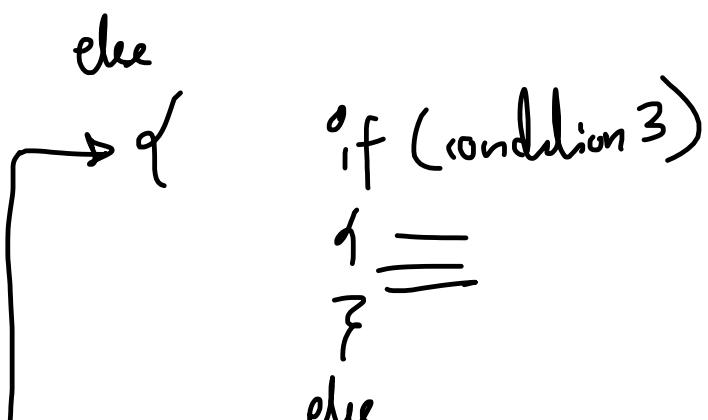
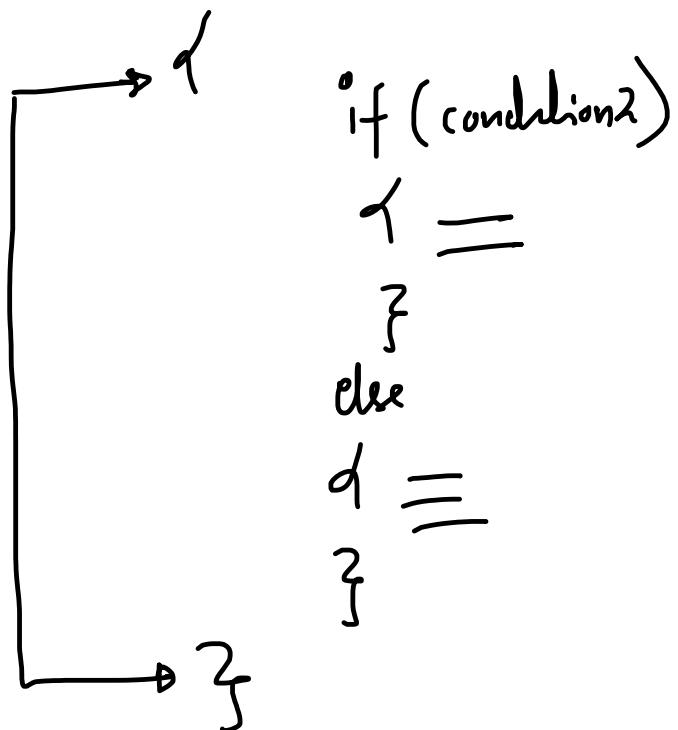
```
L → }     printf("28 days\n");  
else  
    printf("Not valid month\n");
```

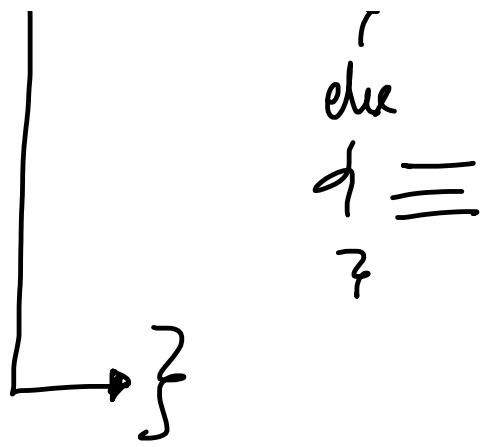
```
#include<stdio.h>  
int main()  
{int m,y;  
printf(" enter the month number\n");  
scanf("%d",&m);  
  
if(m==1 || m==3 || m==5 || m==7 || m==8 || m==10 || m==12)  
printf(" 31 days\n");  
else if(m==4 || m==6 || m==9 || m==11)  
printf(" 30 days\n");  
else if (m==2)  
{  
  
printf("enter the year\n");  
  
scanf("%d",&y);  
  
if( (y%400==0) || (y%4==0 && y%100!=0 ))  
  
printf("29 days\n");  
  
else  
printf("28 days\n");  
}  
else  
printf(" Invalid Month number\n");  
  
return 0;  
}
```

## ④ Nested if - else

- \* If else block can be nested within  
if block or else block
  - \* The Nesting can be done for any no  
of level

Syntax  $\Rightarrow$  if (condition)





Ex

```

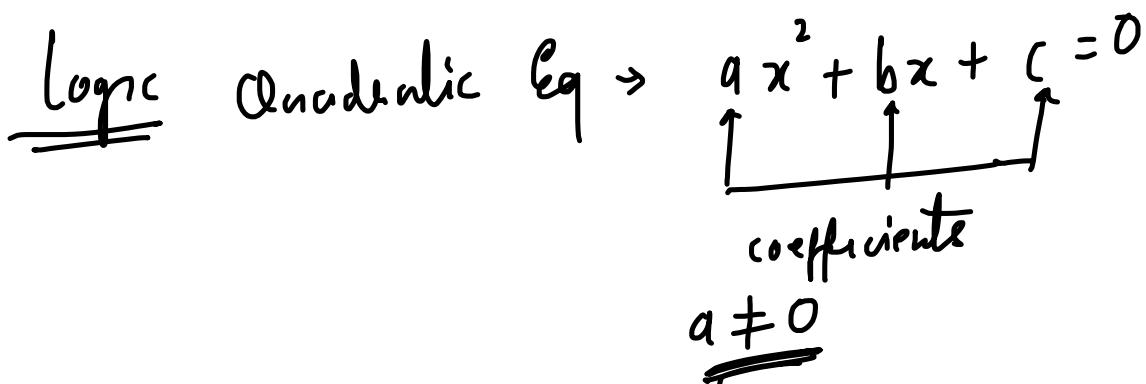
int a, b, c;
scanf("y d y d y.d", &a, &b, &c);
if (a >= b)
{
    if (a > c)
        printf("a is largest\n");
    else
        printf("c is largest\n");
}
else
{
    if (b > c)
        printf("b is largest\n");
    else
        printf("c is largest\n");
}

```

(Ans) a = 10, b = 71, c = 71

Q) C program to read coefficients of Quadratic Eq<sup>n</sup>  
 & display the roots of Equation.

Logic Quadratic Eq  $\rightarrow ax^2 + bx + c = 0$



a  $\neq 0$

Ask user to Enter coefficients

```

float a, b, c, r1, r2;
printf("Enter Coefficients of Equation\n");
scanf("%f %f %f", &a, &b, &c);
if (a == 0)
{
    printf("Eq is not Quadratic\n");
}
else
{
    // a is not zero i.e Eqn is Quadratic.
    Let  $d = b^2 - 4ac,$ 

```

$$\text{let } d = \underline{D - \gamma^2}$$

$$x_1 = \frac{-b + \sqrt{d}}{2a} \quad x_2 = \frac{-b - \sqrt{d}}{2a}$$

Note > if  $d > 0$  (is +ve) ✓  
& then roots  $x_1$  &  $x_2$  are distinct and real

} apply formula

else if  $d == 0$  ✓

& then roots  $x_1$  &  $x_2$  are equal and real

apply formula

}

else

$d$

$d < 0$

} print "Roots are Imaginary"

}

} else

```

#include<stdio.h>
#include<math.h>
int main()
{
    float a,b,c,x1,x2,d; //declare every variables at the start
    printf("Enter the coefficients of quadratic equation\n");
    scanf("%f %f %f",&a,&b,&c);

    if(a==0)
    {
        printf("entered coefficients does not form quadratic equation\n");
    }
    else
    {

        printf("entered coefficients forms quadratic equation\n");
        d=(b*b)-(4*a*c);

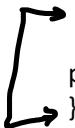
        if(d>0)
        {
            printf("roots are real and distinct\n");
            x1=(-b+sqrt(d))/(2*a);
            x2=(-b-sqrt(d))/(2*a);
            printf("Roots of the quadratic eqns are %f and %f",x1,x2);
        }

        else if(d==0)
        {
            printf("roots are real and equal\n");
            x1=-b/(2*a);
            x2=x1;
            printf("Roots of the quadratic eqns are %f and %f",x1,x2);
        }

        else
            printf("roots are imaginary\n");
    }//outer else

    return 0;
}//MAIN

```



④ Read the no of unit Consumed and Calculate the total

amount to pay      Base price = Rs 100 ✓

Constraints ⇒ Unit      1 — 200  
charge = 80 paise per unit.

200 — 300  
charge = 90 paise per unit.

Unit > 300  
charge = 1 Rs per unit

If total Amount Exceeds Rs 400 then  
additional Surcharge of 15%  
is to be paid.

```
#include<stdio.h>
int main()
{
    float pay=0.0,total,unit;
    printf("enter nos of unit consumed\n");
    scanf("%f",&unit);

    if(unit>=1 && unit<=200)
        pay=0.80*unit;

    else if(unit>200 && unit<=300)
        pay=0.90*unit;

    else if(unit>300)
        pay=1.00*unit;

    total=100+pay;

    if(total>400)
        total=total+(0.15*total);
```

```
printf("total payment to be done= %f\n",total);
return 0;
}
```

⑤ Switch Case → It is similar to else if ladder  
but the Exp of Condition must only  
test for equality and variable must  
be integer or character

General form → To check Kaha Hai

switch(Exp)  
{  
case value1:  
=====  
=====  
=====  
break;

Case value 2 :  
=====  
=====  
=====  
break;

Case value 3 :  
=====  
=====  
=====  
break;

default :  
=====  
break;

}

||End of Switch.

Using Switch

#include <stdio.h>

#include <stdio.h>

```

#include <stdio.h>
int main()
{
    int n;
    printf("Enter the number\n");
    scanf("%d", &n);
    if (n == 1)
        printf("One Hai\n");
    else if (n == 2)
        printf("Two Hai\n");
    else if (n == 3)
        printf("Three Hai\n");
    else
        printf("Kaun Hai\n");
    return 0
}

```

→ #include <stdio.h>

```

int main()
{
    int n;
    printf("Enter the number\n");
    scanf("%d", &n);
    switch(n)
    {
        case 1 : printf("One Hai\n");
                    break;
        case 2 : printf("Two Hai\n");
                    break;
        case 3 : printf("Three Hai\n");
                    break;
        default : printf("Kaun Hai\n");
                    break;
    }
    return 0
}

```

```

#include<stdio.h>
int main()
{
    int ch,a,b;
    printf("Menu:\n");
    printf("1:sum\n2:diff\n3:product\n4:quotient\n");
    printf("enter two integer values\n");
    scanf("%d%d",&a,&b);
    printf("enter the choice\n");
    scanf("%d",&ch);

    switch(ch)

```

```
{  
case 1:printf("sum= %d\n",a+b);break;  
case 2:printf("diff= %d\n",a-b);break;  
case 3:printf("product= %d\n",a*b);break;  
case 4:printf("quotient= %d\n",a/b);break;  
default:printf("invalid choice\n");  
}  
return 0;  
}
```

## Fallthrough in Switch Case →

if  $\ominus$  in a switch block, break is not specified for a particular case, and if the case happens to be true then that case  $\ominus$  is Executed and following cases are also executed until break is encountered or switch block is terminated

Eg

```
#include<stdio.h>
```

```
int main()
{
    int x;
    printf("enter value\n");
    scanf("%d",&x);
    switch(x)
    {
        case 1:printf("one hai\n");
        case 2:printf("two hai\n");
        case 3:printf("three hai\n");break;
        default:printf("kaun hai\n");break;
    }
    return 0;
}
```

not mandatory to specify  
 → break for default case at the  
end

D:\ODD 2021-22\SPA INFT\SPA PROGRAMS\fallthrough.exe

```
enter value
1
one hai
two hai
three hai
```

Process exited after 2.583 seconds with return value 0
 Press any key to continue . . .

Q) Write a program to read month no from user & display no of days in the month (Using switch case & fall through)

```
#include<stdio.h>
int main()
{
    int n,y;

    printf("enter month no\n");
    scanf("%d",&n);
    switch(n)
    {
        case 1:
        case 3:
        case 5:
        case 7:
        case 8:
        case 10:
        case 12:printf("31 days\n");break;

        case 4:
        case 6:
        case 9:
        case 11:printf("30 days\n");break;

        case 2:printf("enter the year\n");
                 scanf("%d",&y);

                 if( (y%400==0) || (y%4==0 && y%100!=0 ))
                     printf("29 days\n");
                 else
                     printf("28 days\n");
                 break;

        default:printf("invalid month number\n");break;
    }//switch over
    return 0;
}
```

Q) Program to read a character from user and determine it is vowel or consonant

```
#include<stdio.h>
int main()
{
    char ch;
    printf("enter the character\n");
    scanf("%c",&ch);
    //upper case character lower case characters
    // if((ch>='A' && ch<='Z') || (ch>='a' && ch <= 'z'))
    if((ch>=65&&ch<=90)|| (ch>=97&&ch<=122))
    {
        switch(ch)
        {
            case 'A':
            case 'a':
            case 'E':
            case 'e':
            case 'i':
            case 'I':
            case 'o':
            case 'O':
            case 'u':
            case 'U':printf("It's vowel\n");break;
            default:printf("It's consonant\n");
        }
    }
    else
        printf("It's not an alphabet\n");
}
return 0;
```

*→ character literal in single quote*

Q) WAP to calculate following.

Emp  
Grade

A  $\rightarrow$  Rs 100/hr for regular hours & no payment for overtime hours

B }  $\rightarrow$  Rs 50/hr for regular hours & 1.5 time for overtime hours

C D  $\rightarrow$  Rs 30/hr for regular hours & 1.75 time for overtime hours upto regular hours, & 2 time for overtime hours greater than regular hours

(calculate One day payment !!)

Let  $\Rightarrow$       req      overtime      pay  
Grade A      8 hours      10 hours       $8 \times 100 + 0 \times 10 = 800$   
 $\downarrow$   
 $\boxed{\text{pay} = 100 \times \underline{\text{req}}}$

Grade B /      8 hours      6 hours       $\Rightarrow 50 \times 8 + 1.5 \times 50 \times 6$

Grade C      8 hours      10 hours       $\Rightarrow 50 \times 8 + 1.5 \times 50 \times 10$   
 $\downarrow$   
 $\boxed{\text{pay} = 50 \times \underline{\text{req}} + 50 \times 1.5 \times \underline{\text{overt}}}$

Grade D

$$\text{pay} = \underline{50 \times \underline{\text{req}}} + \underline{50 \times 1.5 \times \underline{\text{over}}}$$

$\underline{\text{req}}$   
8 hours

$\underline{\text{over}}$   
4 hours  
=  $8 \times 30 + 1.75 \times 30 \times 4$

18 hours { 8 hours

10 hrs  $\Rightarrow \underline{30 \times 8} + \underline{1.75 \times 30 \times 8}$

8  $\underline{\underline{\text{reg}}} > \underline{\underline{\text{req}}}$   $+ 2 \times 30 \times (10 - 8)$

if (over > req)

{  
 pay =  $30 \times \underline{\underline{\text{req}}} + 1.75 \times 30 \times \underline{\underline{\text{over}}} + 2 \times 30 \times (\underline{\underline{\text{over}}} - \underline{\underline{\text{req}}})$ ;  
 }  
 else  
 pay =  $30 \times \underline{\underline{\text{req}}} + 1.75 \times 30 \times \underline{\underline{\text{over}}}$ ;       $\frac{\text{req}}{8} \quad \frac{\text{over}}{8}$

```
#include<stdio.h>
```

```
int main()
{
char grade;
float reg, over, pay;

printf("enter the grade\n");
scanf("%c", &grade);

printf("enter regular and overtime hours\n");
scanf("%f %f", &reg, &over);

switch(grade)
{
case 'a':
case 'A':
```

```

pay=100*reg;
printf("payment = %f\n",pay);
break;

case 'b':
case 'c':
case 'B':
case 'C': pay=50*reg+1.5*50*over;
printf("payment = %f\n",pay);
break;

case 'd':
case 'D': if(over>reg)
    pay=30*reg+1.75*30*reg+2*30*(over-reg);
else
    pay=30*reg+1.75*30*over;
printf("payment = %f\n",pay);
break;

default:printf("invalid grade\n");
}
return 0;
}

```

outende switch.

Using Else if ladder →

```

#include<stdio.h>

int main()
{
char ch;
float reg,over,pay;

printf("enter the grade\n");
scanf("%c",&ch);

printf("enter regular and overtime hours\n");
scanf("%f %f",&reg,&over);

if(ch=='A' | ch=='a')
{
    pay=100*reg;
    printf("payment = %f\n",pay);
}

```

```
else if( ch=='B' || ch=='b' || ch=='C' || ch=='c')
{
    pay=50*reg+1.5*50*over;
    printf("payment = %f\n",pay);
}

else if(ch=='D' || ch=='d')
{
    if(over>reg)
        pay=30*reg+1.75*30*reg+2*30*(over-reg);
    else
        pay=30*reg+1.75*30*over;
    printf("payment = %f\n",pay);
}
else
printf("invalid grade\n");

return 0;
}
```

## Fullthrough Revision

```

int n;
printf("Enter the no\n");
scanf("%d", &n);
switch(n)
{
    case 1: printf("One\n");
    case 2: printf("Two\n"); break;
    case 3: printf("Three\n"); break;
    default: printf("Default\n");
}
printf("Outside Switch block\n");

```

" When break is executed  
in switch block the  
control is transferred  
outside the switch block  
and all the following  
cases are bypassed "

Case 1) No break after  
case 1  
n=1  
Output  
One  
Two  
Outside Switch block

Case 2) No break after  
case 1

n=1  
One  
Two  
Outside Switch block

## Consider

```

switch(n) Case label
{
    case 1: printf("One\n");
    case 2: printf("Two\n");
}

```

No break after case 1  
& case 2

(1) n=2  
Two  
Three

case 3 printf(" Three\n"); break  
default: printf(" Default\n"); break;

⑪ n=1  
One  
Two  
Three

It is known as fallthrough b'coz the control falls from One case label to another

Menu Driven

<u>Item Code</u>	<u>Item Name</u>
1	AC

Pricing

```

price > 30K
dis = 50%
price > 20K
dis = 40%
else
dis = 10%.

```

2 Refrigerator

```

price > 25K
dis = 20%
price > 10K
dis = 10%
else
no discount

```

3 Oven

```

price > 15K
dis = 32%
price > 8K
dis = 25%
else
dis = 5%

```

\*

Read Item Code & price from user

\* Display the amount to ~~pay~~

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

```

{ int c;
float price,dis;

printf("MENU:\n");
printf("1:AC\n2:Refrigerator\n3:Oven\n");
printf(" enter item code\n");
scanf("%d",&c);
printf(" Enter Price\n");
scanf("%f",&price);

switch(c)
{
case 1: if(price>30000)
    dis=0.5*price;
    else if(price>20000)
    dis=0.4*price;
    else
    dis=0.1*price;
    printf(" Amount to pay= %f\n",price-dis); -->
    break;
case 2: if(price>25000)
    dis=0.2*price;
    else if(price>10000)
    dis=0.1*price;
    else
    dis=0;
    printf(" Amount to pay= %f\n",price-dis); -->
    break;
case 3:if(price>15000)
    dis=0.32*price;
    else if(price>8000)
    dis=0.25*price;
    else
    dis=0.05*price;
    printf(" Amount to pay= %f\n",price-dis); -->
    break;
default: printf(" invalid item code\n");
}//SWITCH ENDS
return 0;
}

```

$\text{if } (c == 1 \text{ || } c == 2 \text{ || } c == 3)$   
 $\text{printf} (" \text{Amount to Pay} = %.f \text{ }\n",$   
 $\text{price-dis});$

Q) Read 2 floating point values and check if they are equal upto 2 decimal places after floating point

```
#include<stdio.h>
int main()
{
    float x;
    float y;
    int x1, y1;
    printf("enter value of x and y\n");
    scanf("%f %f", &x, &y);
    x1 = (int)(x * 100); type casting
    y1 = (int)(y * 100);
    if(x1 == y1)
        printf("nos are equal upto 2 places after floating point\n");
    else
        printf("nos are not equal upto 2 places after floating point\n");
    return 0;
}
```

(Ans 1)

$$\begin{aligned} x &= 3.5 \underline{2} \overset{+1}{\cancel{6}} \\ y &= 3.5 \underline{3} \overset{+2}{\cancel{7}} \end{aligned}$$

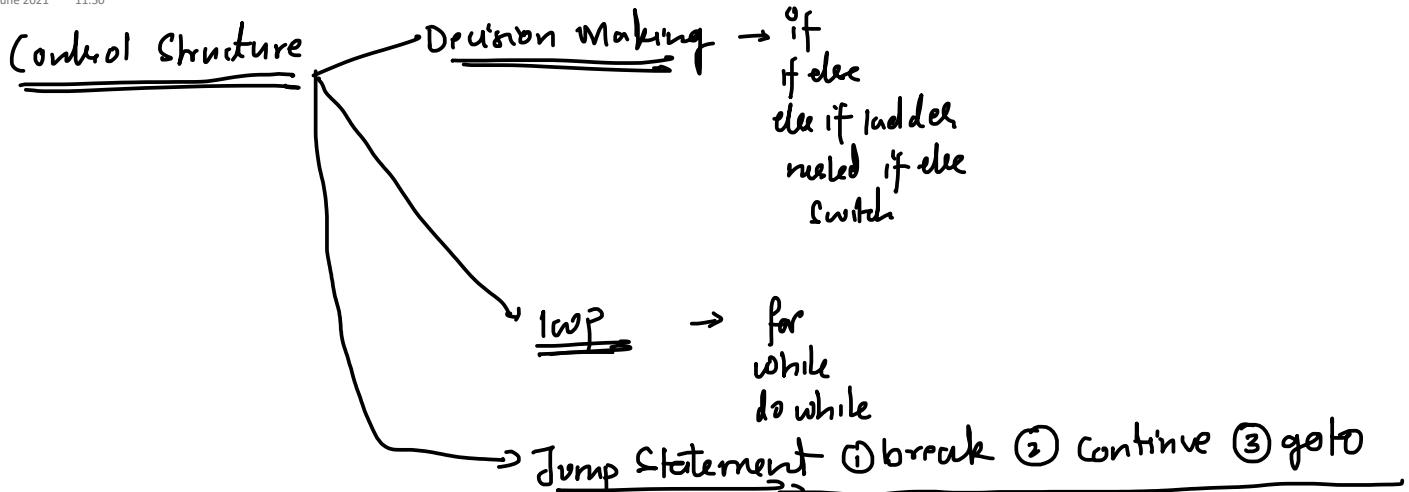
x1 = (int) 352 61  
y1 = (int) 353 71

(Ans 2)

$$\begin{aligned} x &= 3.5 \overset{+1}{\cancel{2}} \overset{+6}{\cancel{1}} \\ y &= 3.5 \overset{+1}{\cancel{2}} \overset{+7}{\cancel{1}} \end{aligned}$$

x1 = (int) 352.61  
y1 = (int) 352.71

Note → for upto 3 decimal places after floating point multiply with 1000 and so on



loop in C \* There is certain piece/block of code that is supposed to be executed repeatedly for certain no of times, then piece/block of code can be placed within a loop  
\* loops are iterative in nature

### ① for loop

Syntax → *(<sup>initializer / counter</sup><sub>is initialized</sub>)* *Terminating cond'n* *Decides no of times the loop iterates.*  
 $\text{for}(\underline{\text{initialization}}; \underline{\text{condition}}; \underline{\text{expression}})$   
 1 Block of Code      \* Initialization of "iterator" is done only once

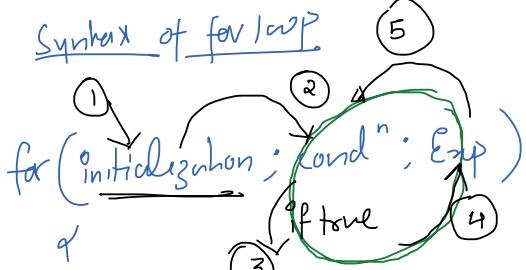
Ex *(Case 1) without { }*  
Print Hello 5 times

{ printf("Hello\n");  
 printf("Hello\n");  
 printf("Hello\n");  
 printf("Hello\n");  
 printf("Hello\n"); } Redundant Code

(Case 2) with for loop  
 $\text{int } i; \quad \text{(iterator)}$   
 $\text{for}(\underline{i=1}; \underline{i \leq 5}; \underline{i=i+1})$   
 1 printf("Hello\n");  
 {  
 }  
 223456  
 =  
 a byte

$i=1$ $i=2$ $i=3$ $i=4$ $i=5$	<b>if</b> Hello Hello Hello Hello Hello
---	--

### Syntax of for loop



If true  
 Body of for loop  
 is Executed once

(case 2) can be rewritten as

① `int i;`  
`for(i=1; i<=5; i++)`  
`printf("Hello\\n");`

② `int i;`

`for(i=1; i<=5; i++)`  
`printf("Hello\\n");`

③ `int i=1;`

`for(; i<=5; i++)`  
`printf("Hello\\n");`

④

`int i=1;`  
`for( ; i<=5; )`  
`{ printf("Hello\\n");`  
`i++;`  
`}`

\* To execute the loop 5 time →

`int i;`  
`for(i=1; i<100; i=i+20)`  
`printf("Hello\\n");`

$i=1$	- Hello
$i=21$	- Hello
$i=41$	- Hello
$i=61$	- Hello
$i=81$	- Hello
$\underline{i=101}$	NO

Or `int i,`  
`for(i=5; i>=1; i--)`  
`printf("Hello\\n");`

$i=5$	- Hello
$i=4$	- Hello
$i=3$	- Hello
$i=2$	- Hello
$i=1$	- Hello
$i=0$	X

Nesting of for loop → \* for loop can be nested within another for loop

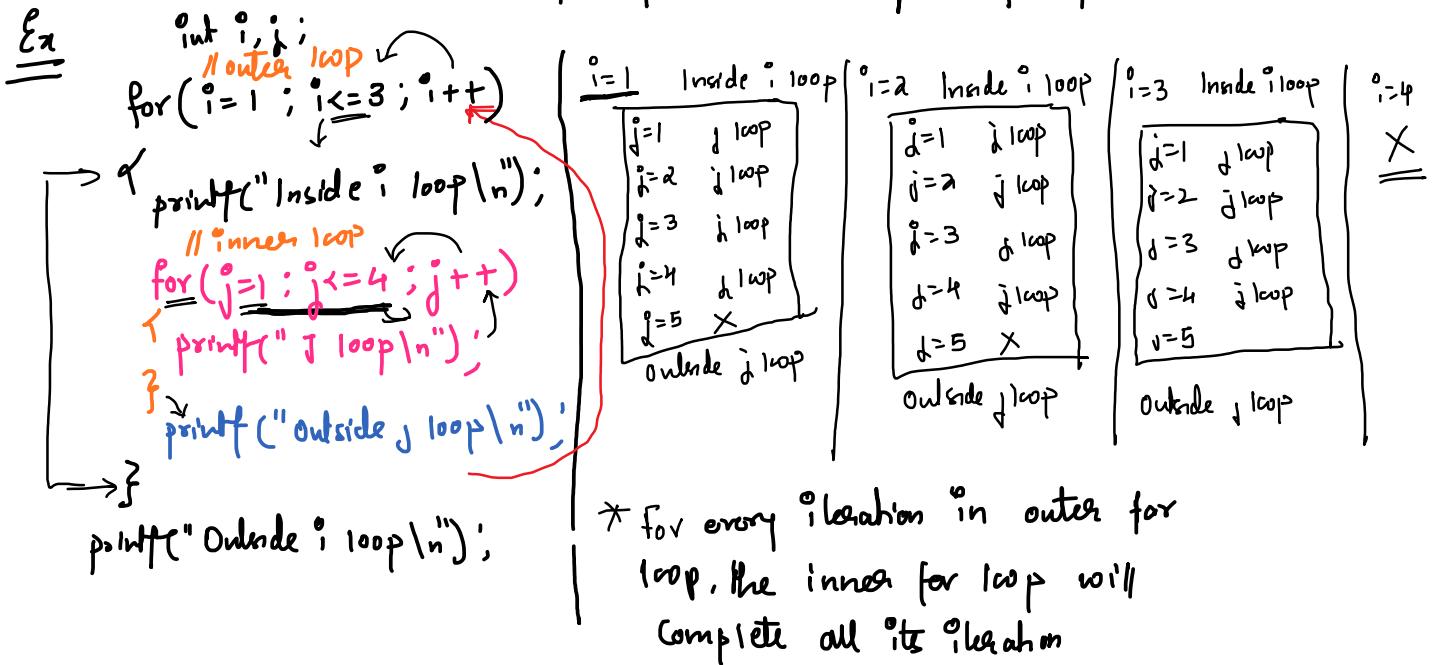
\* The nesting can be done for any number of levels.

\* keep different operator for diff loop

Ex

`int i, j;`  
`for(i=1; i<=100; i++)`

“keep different iterator for diff loop”



## (Jump Statement)

break statement in loop =

- \* break stmt when executed in switch block will take the control outside switch block
- \* When break statement is executed in loop, it will terminate the current loop and control is transferred to the first statement outside the current loop
- \* No further iterations are executed.

Eg

```

int i,
for( ; i<=10 ; i++)
{
    printf("Hi\n"),
    if(i>=3)
        * break;
    printf("Hello\n");
}
printf("Outside for loop\n");

```

i=1      Hi  
         Hello  
         i=2      Hi  
         Hello  
         i=3      Hi  
             Outside for loop.

Eg

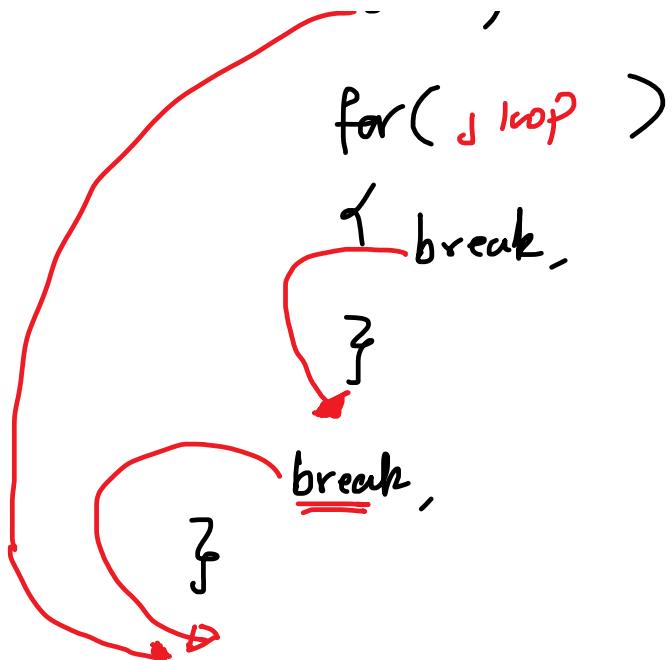
```
for ( ; loop )
```

```

{
    break;
}

```

break will take control outside the loop (current)



outside the loop (current)  
i.e. loop in which  
break is executed

## Continue statement

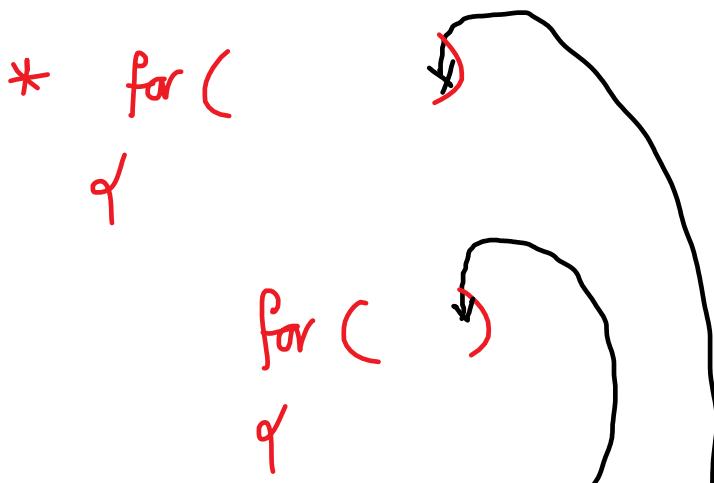
\* When continue statement is executed in loop  
 It terminates that iteration of current loop and control is transferred to the next iteration of current loop.

Ex    int  $\theta$ ;

```

for( $i=1$  ;  $i \leq 5$  ;  $i++$ )
{
    printf("Hi\n");
    if ( $i >= 3$ )
        continue;
    printf("Hello\n");
}
printf("Outside loop\n");
    
```

$i=1$ $i=2$ $i=3$ $i=4$ $i=5$ $i=6$	Hi Hello Hi Hello Hi 
--	--------------------------------------



q  
continue';  
}  
q  
continue';  
}

# Menu Driven Arithmetic Calculator Using $f^n$ for following Operations

1' Addition

2' Subtraction

3' Multiplication

4' Division

5: Remainder

```
#include<stdio.h>
```

```
int add(int x,int y) → fn Defn
{
    return x+y;
}
```

```
int subs(int x,int y) → fn Defn
{
    return x-y;
}
```

```
int mul(int x,int y) → fn Defn
{
    return x*y;
}
```

```
int div(int x,int y) → fn Defn
{
    return x/y;
}
```

```
int mod(int x,int y) → fn Defn
{
    return x%y;
}
```

```
int main()
{
```

```

int ch;
int a,b;

do
{

printf("\n1:Addition\n2:Substraction\n");
printf("3:Multiplication\n4:Division\n");
printf("5:Remainder\n6:exit\n");

printf("enter two number\n");
scanf("%d %d",&a,&b);

printf("enter choice\n");
scanf("%d",&ch);

switch(ch)
{
    case 1:printf("sum= %d\n",add(a,b));break;
    case 2:printf("diff= %d\n",subs(a,b));break;
    case 3:printf("product= %d\n",mul(a,b));break;
    case 4:printf("quotient= %d\n",div(a,b));break;
    case 5:printf("remainder= %d\n",mod(a,b));break;
    case 6:printf("exit condition encountered\n");break;
    default:printf("invalid choice\n");break;
}
}while(ch!=6);
return 0;
}

```

\* Number of formal args Specified in fn Definition  
Must Match to the no of actual args passed in fn

(all)

\* Corresponding data types of formal args and actual args must match

float add (int x, float y)  $\Rightarrow$  F<sup>n</sup> Def<sup>n</sup>  
 {  
 return x+y;  
 }

- ① ~~float~~ r = add (10.5, 20.5); X
- ② float r = add (10, 20.5); ✓
- ③ float r = add (10, 20); ✓

Call By Value  $\rightarrow$  VImp

Consider Ex ->

```
#include<stdio.h>
```

void f1(int,int); // F<sup>n</sup> Decl<sup>n</sup>

int main()  
{  
int x=10,y=20;  
     $\rightarrow$  local

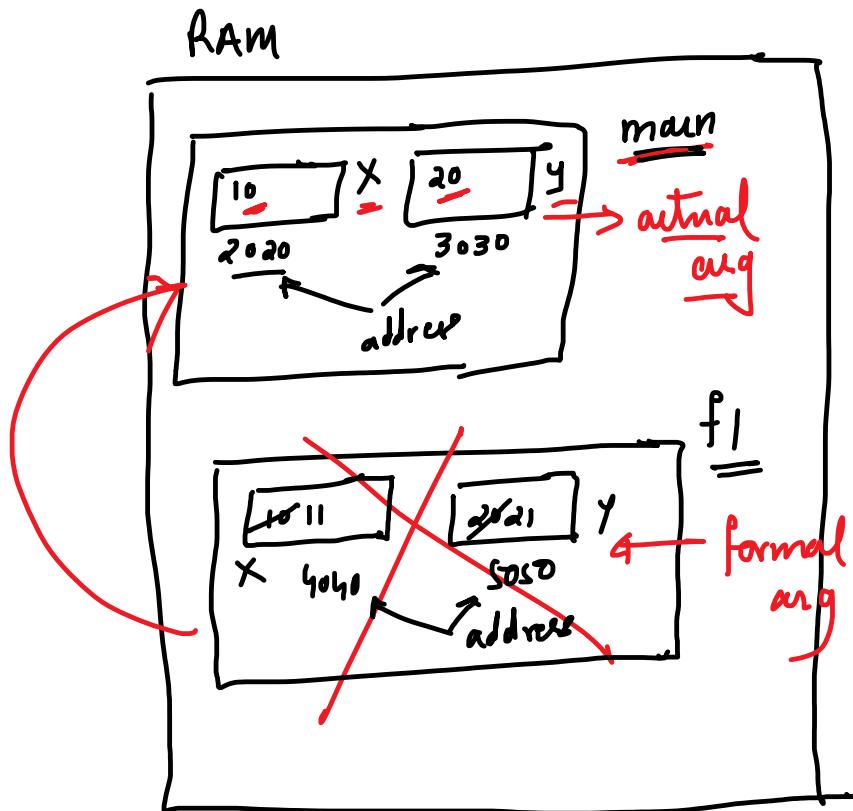
printf("before function call\n");  
printf("x= %d\n",x);  
printf("y= %d\n",y);  
f1(x,y); //function call  
 $\downarrow$   
10 20

printf("after function call\n");  
printf("x= %d\n",x);  
printf("y= %d\n",y);  
     $\downarrow$

return 0;  
}

// F<sup>n</sup> Definition  
void f1(int x,int y)  
{  
     $\downarrow$     $\downarrow$     $\rightarrow$  local  
x=x+1;  
y=y+1;

printf("In function f1\n");  
printf("x= %d\n",x);  
printf("y= %d\n",y);  
}



Op before function call

$$x = 10$$

$$y = 20$$

In function f1

$$x = 11$$

$$y = 21$$

after function call

$$x = 10$$

$$y = 20$$

\* In Call By Value, the actual argument passed in function call are values only that gets copied

## into formal arguments at function Definition

- \* Here Memory Referred by formal arg & actual arg are different hence changes made in formal arg (In fn Def) are not reflected in actual arg (fn call → calling fn)

Q10 →

```
Q10 D:\ODD 2021-22\SPA\INFT\SPA PROGRAMS\callbyvalue.exe
before function call
x= 10
y= 20
In function f1
x= 11
y= 21
after function call
x= 10
y= 20
```

---

## Call By Value Example 2 →

```
#include<stdio.h>
void swap(int x,int y)
{
    int t;
    t=x;
    x=y;
    y=t;
}

int main()
{
    int x=10,y=20;

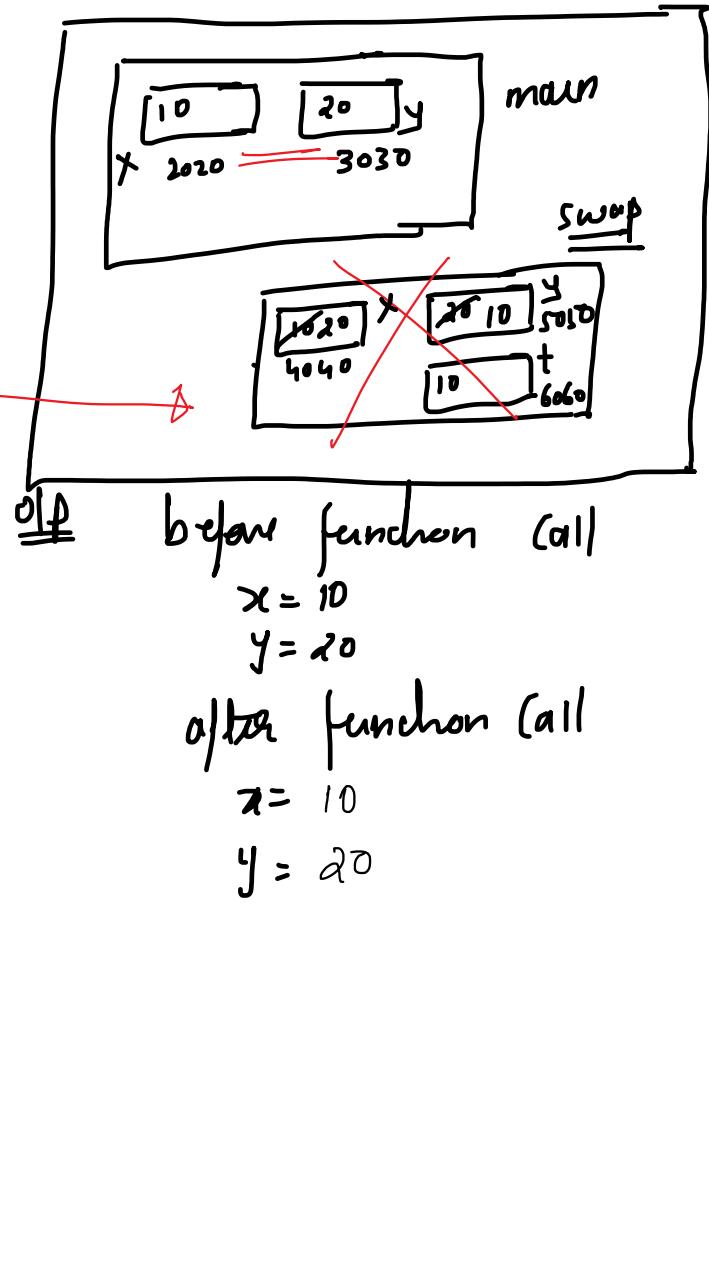
    printf("before function call\n");
    printf("x= %d\n",x);
    printf("y= %d\n",y);

    swap(x,y); //function call
    Call By Value

    printf("after function call\n");
    printf("x= %d\n",x);
    printf("y= %d\n",y);

    return 0;
}
```

RAM



Q) Write a C program to create a function that returns factorial of a Number use this f<sup>n</sup> to find Sum of following Series →

$$\textcircled{1} \Rightarrow \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \dots + \frac{1}{n!} \quad \checkmark \quad \frac{1}{\text{fact}(i)} \\ i=1 \rightarrow n$$

$$\textcircled{2} \quad \frac{1}{1!} - \frac{1}{2!} + \frac{1}{3!} - \frac{1}{5!} \dots + \frac{1}{n!} \quad \checkmark$$

Q)  $\textcircled{1}$  `#include <stdio.h>`

`int fact(int n)`

```
int i, f = 1;
for(i = 1; i <= n; i++)
    f = f * i;
// If stores factorial of n
return f;
```

$$5! = 1 \times 2 \times 3 \times 4 \times 5$$

fact(n)  
i, f = 1

$$\begin{array}{ll} i=1 & f = 1 \times 1 = 1 \\ i=2 & f = 1 \times 2 = 2 \\ i=3 & f = 2 \times 3 = 6 \\ i=4 & f = 6 \times 4 = 24 \\ i=5 & \end{array}$$

`int main()`

```
int i, n;
float sum = 0;
```

↓ `Driver Code` ↓ ↓ ↓ ↓ ↓

printf("Enter the value of n \n");  
 scanf("%d", &n);

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

Sum = Sum + (1.0 / fact(i));  
 ↗ f^n (call)

printf("Sum of Series = %f \n", sum);

→ return 0;

}

$$\begin{aligned}
 & n = 4, \text{sum} = 0 \\
 & i=1 \\
 & \text{sum} = 0 + \underline{1.0 / \text{fact}(1)} = 1.0 \\
 & i=2 \\
 & \text{sum} = 1.0 + (\underline{1.0 / \text{fact}(2)}) \\
 & = 1.0 + (1.0 / 2) = 1.0 + 0.5 \\
 & \Rightarrow 1.5 \\
 & i=3 \\
 & \text{sum} = 1.5 + (1.0 / \text{fact}(3)) \\
 & = 1.5 + (1.0 / 6) \\
 & 1.5 + 0.167 \\
 & = 1.667 \\
 & i=4 \\
 & \text{sum} = 1.667 + (1.0 / \text{fact}(4)) \\
 & = 1.667 + (1.0 / 24) \\
 & = 1.667 + 0.041 \\
 & = 1.708
 \end{aligned}$$

② #include <stdio.h>

int fact(int n)  
 { int i, f=1;  
 for(i=1; i<=n; i++)  
 f=f\*i;  
 return f;  
 }

int main()  
 { int n; cin >> n; cout << fact(n); }

$$\begin{aligned}
 & n = 4, \text{sum} = 0, \text{sign} = 1
 \end{aligned}$$

in main

```
1 int i, n, sign=1;
float sum=0;
printf("Enter the value of n\n");
scanf("%d", &n);
for(i=1; i<=n; i++)
{
    sum = sum + (1.0 / fact(i)) * sign;
    sign = sign * -1;
}
printf("Sum of Series = %f\n", sum);
return 0;
```

}

n=k      sum=0, sign=1

i=1

$$\begin{aligned} \text{sum} &= 0 + (1.0 / \text{fact}(1)) * \underline{\underline{1}} \\ &= 0 + 1.0 = \underline{\underline{1.0}} \end{aligned}$$

sign = -1

i=2

$$\begin{aligned} \text{sum} &= 1.0 + (1.0 / \text{fact}(2)) * \underline{\underline{-1}} \\ &= 1.0 - (1.0 / \text{fact}(2)) = \underline{\underline{0.5}} \end{aligned}$$

### ③ Sine Series

$$\sin(x) = \frac{x^1}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots$$

### ④ Cosine Series

$$\begin{aligned}\cos(x) &= 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots \\ &= \frac{x^0}{0!} - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots\end{aligned}$$

Sol

③ #include <stdio.h>

Sine Series

$\text{int fact(int n)}$

```

    {
        int i, f=1;
        for (i=1; i<=n; i++)
            f=f*i;
    }
```

return f;

$\text{int main()}$

```

    {
        int i, n, sign=1;
        float x, sum=0.0;
```

```

printf("Enter no of terms\n"),
scanf("%d", &n);

printf("Enter the angle in degree\n"),
scanf("%f", &x);

x = (x * 3.14) / 180; // into radian

t = x;
for(i=1; i<=n; i++)
{
    sum = sum + (t / fact(i)) * sign;
    sign = sign * -1;
    t = t * x * x;
}
printf("Sum of Given sine Series = %f\n", sum);
return 0;
}

```

⑤ #include <stdio.h>      Cosine Series

```

int fact(int n)
{
    int i, f = 1;
}

```

$\text{int fact(int } n\text{)}$

$\rightarrow \left\{ \text{int } i, f = 1;$

```

    ↗ i = 1; f = 1;
    for (i = 1; i <= n; i++)
        f = f * i;
    return f;
}

```

```

int main()
{
    int i, n, sign = -1;
    float x, sum = 1, t,

```

```

printf("Enter no of terms\n");
scanf("%d", &n);

```

```

printf("Enter the angle in degree\n");
scanf("%f", &x);

```

$$x = (x \times 3.14) / 180; // \text{into radians}$$

$$t = x * x;$$

```

for (i = 1; i <= n; i++)
{
    sum = sum + (t / fact(2 * i)) * sign;
    sign = sign * -1;
    t = t * x * x;
}

```

$$\begin{aligned}
& i = 1 \\
& sum = 1 + (x^2 / fact(2)) * -1 \\
& = 1 - \frac{x^2}{fact(2)}
\end{aligned}$$

```
printf("Sum of Given cosine Series = %.f\n", sum);  
return 0;  
}
```

## goto Statement

- \* It is a jump statement
- \* It transfers the flow of control abruptly to specified location in the program

Syntax →

`goto labelname;`

- \* label are the checkpoints/milestones assigned to the code at specific pos"

Consider

```
// Outer for loop
for( i=1 ; i<=5 ; i++ )
{
    if( i>=3 )
        break;
    // Inner for loop
    for( j=1 ; j<=4 ; j++ )
        if( j>=2 )
            break;
}
```

Consider →

```
L1: for( i=1 ; i<=5 ; i++ )
    {
        if( i>=3 )
            goto L1;
    }

L2: for( j=1 ; j<=4 ; j++ )
    {
        if( j>=2 )
            goto L3;
    }

L3: printf("Hello");

```

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

D:\ODD 2021-22\SPA INFT\SPA PROGRAMS\c1.exe  
Hello  
hi

```

int main()
{
    printf(" hello\n");
    printf(" hi\n");
    goto l1; (takes the control to the  

label -l1)
    printf(" kaise ho\n");
    printf(" theek hoon\n");
l1: printf(" study karo\n");
    printf(" apna kaam karo\n");
    return 0;
}

```

D:\ODD 2021-22\SPA INFT\SPA PROGRAMS\c1.exe

hello  
hi  
study karo  
apna kaam karo

---

Note →

~~int i, j;~~

More than one ~~initializer~~ initialization

for(~~i=1, j=2~~; ~~i < j~~; ~~i++, j++~~)

of ~~i = 1, j = 2~~  
~~i = 1, j = 2~~  
~~i = 1, j = 2~~  
~~i = 1, j = 2~~

$\Phi = \Phi_1 \times \Phi_2$

More than one ~~initializer~~  $\downarrow$  Exp.

} More than one ~~initializ<sup>n</sup>~~ possible  $\rightarrow$  Separated with Comma

More than one Exp possible  $\rightarrow$  Separated with Comma -

Q1) WAP to display sum of first n natural no

logic let  $n=5$ ,  $\underline{\underline{sum}} = \underline{\underline{0}}$ ;

$$\left\{ \begin{array}{ll} i=1, & sum = sum + \underline{\underline{i}} \Rightarrow 0 + 1 = \underline{\underline{1}} \\ i=2 & \Rightarrow sum = sum + \underline{\underline{i}} \Rightarrow 1 + 2 = \underline{\underline{3}} \\ i=3 & sum = sum + \underline{\underline{i}} \Rightarrow 3 + 3 = \underline{\underline{6}} \\ i=4 & sum = sum + \underline{\underline{i}} \Rightarrow 6 + 4 = \underline{\underline{10}} \\ i=5 & sum = sum + \underline{\underline{i}} \Rightarrow 10 + 5 = \underline{\underline{15}} \end{array} \right.$$

#include <stdio.h>

int main()

{ int n, i, sum=0;

printf("Enter the value of n\n");  
scanf("%d", &n);

for (i=1 ; i<=n ; i++)  
sum = sum + i;

printf("Sum of first %d natural no is %d\n", n, sum);

} return 0;

Q2) Write a C program to display factorial of n

Logic  $n! = \text{Product of all natural nos from 1 to } n$

$$5! = \frac{5 \times 4 \times 3 \times 2 \times 1}{(1 \times 2 \times 3 \times 4 \times 5)}$$

```
#include<stdio.h>
int main()
{
    int i,n,f=1;

    printf("enter value of n\n");
    scanf("%d",&n);

    for(i=1;i<=n;i++)
        f=f*i;      5       ↗

    printf("factorial of %d = %d\n",n,f);
    return 0;
}
```

$$\underline{\underline{i}} \quad n=5, f=1$$

$$i=1 \quad f=f \times 1 = 1 \times 1 = \underline{\underline{1}}$$

$$i=2 \quad f=\underline{f} \times 2 = 1 \times 2 = \underline{\underline{2}}$$

$$i=3 \quad f=f \times 3 = 2 \times 3 = \underline{\underline{6}}$$

$$i=4 \quad f=f \times 4 = 6 \times 4 = \underline{\underline{24}}$$

$$i=5 \quad f=f \times 5 = 24 \times 5 = \underline{\underline{120}}$$

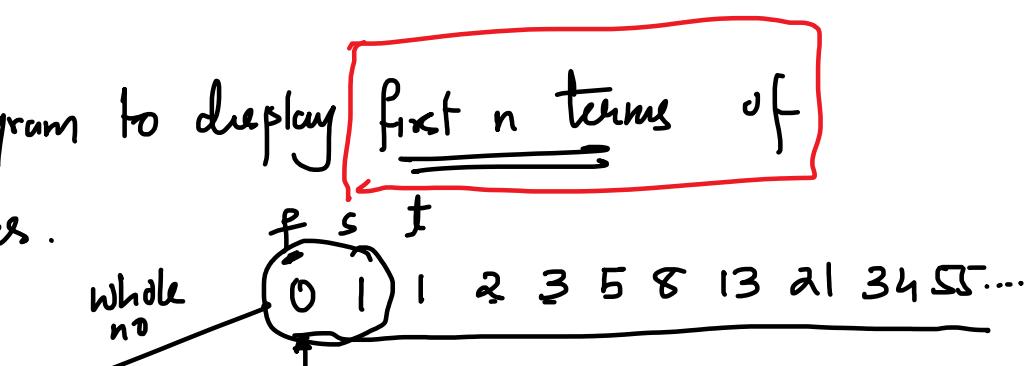
$$\underline{\underline{i}}=6 \quad x$$

Q3) Write a C program to display first n terms of fibonacci Series.

Logic Fibonacci Series

Every 3<sup>rd</sup> term  
of Series is

Sum of previous  
2 terms



Natural no  
(unless until  
specified go with natural no)

α Terms

Specify go with maxima ..

Now let  $n=7$  (Display first 7 (Seven) terms of fibo Series)

1	1	2	3	5	8	13
f	s	t	t	t	t	t
f	s	t	t	t	t	t
f	s	t	t	t	t	t
f	s	t	t	t	t	t

first 2 terms are fixed f,s

7 terms

for 5 times ( $n-2$ ) times  
Calculate 3rd term .

```
#include<stdio.h>
int main()
{
    int n,i,f=1,s=1,t;

    printf("enter nos of terms\n");
    scanf("%d",&n);

    printf("first %d terms of fibo series are\n",n);

    if(n==1)
        printf("%d\n",f);

    else if(n>=2)
    {
        printf("%d\t%d\t",f,s); // first 2 terms f &s are displayed
        for(i=1;i<=n-2;i++)
        {
            t=f+s;
            printf("%d\t",t);
            f=s;
            s=t;
        }
        printf("\n");
    }
    else
        printf("invalid no of terms\n");

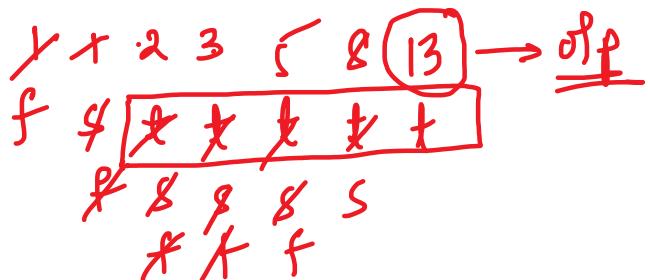
    return 0;
}
```

}

Q4) write a C program to display  $n^{\text{th}}$  term of Fibonacci Series

Ex     $n=7$

o/p



if  $n=7$

we have to display 5<sup>th</sup> third term

```
#include<stdio.h>
int main()
{
    int n,i,f=1,s=1,t;

    printf("kaunsa terms\n");
    scanf("%d",&n);

    printf("the %d terms of fibo series is\n",n);

    if(n==1)
        printf("%d\n",f);

    else if(n==2)
        printf("%d\n",s);

    else if(n>2)
    {

        for(i=1;i<=n-2;i++)
        {
            t=f+s;
            f=s;
            s=t;
        }
        printf("%d\n",t);
    }
}
```

```
}

else
printf("invalid term\n");

return 0;

}
```

Q5) WAP program to determine a number  $n$  is prime or not?

Logic

prime no  $\rightarrow$  A no that gets divisible by 1 and by itself and by no other value

$n=11$  Divide  $n$  by all the values from 1 to  $n(11)$

$n=11$  will be completely divisible only 2 times

$$\underline{0_1=1} \text{ and } \underline{0_2=11}$$

$n=10$  divide  $n=10$  by all values from 1 to 10

if it is divisible by  
by 1  
by 2  
by 5  
by 10 } 4 times

$n=9$  divide  $n=9$  by all values from 1 to 9

it is divisible by  
1 } 3 times  
3 }  
9 }

to get divided only 2 times from 1 to  $n$

Approach A No is

prime if

it not get divided from 2 to  $n-1$

```

#include<stdio.h>

int main()
{
    int n,i,c=0;

    printf("enter the number\n");
    scanf("%d",&n);

    // divide n by all values from 1 to n
    for(i=1;i<=n;i++)
    {
        if(n%i==0)//count how many times it gets
            completely divided
        c++;
    }

    if(c==2)
        printf("%d is a prime number\n",n);
    else
        printf("%d is not a prime number\n",n);
}

```

return 0;

$$n=7, c= \cancel{0} + 2$$

$$i=1 \quad 7 \div 1 = 0 \text{ Yes}$$

$$i=2 \quad 7 \div 2 = 0 \text{ NO}$$

$$i=3 \quad 7 \div 3 = 0 \text{ NO}$$

$$i=4 \quad 7 \div 4 = 0 \text{ NO}$$

$$i=5 \quad 7 \div 5 = 0 \text{ NO}$$

$$i=6 \quad 7 \div 6 = 0 \text{ NO}$$

$$i=7 \quad 7 \div 7 = 0 \text{ Yes}$$

$n=7$  is prime

$$n=8 \quad c=\cancel{0} + 2$$

$$i=1 \quad 8 \div 1 = 0 \text{ Yes}$$

$$i=2 \quad 8 \div 2 = 0 \text{ Yes}$$

$$i=3 \quad 8 \div 3 = 0 \text{ NO}$$

$$i=4 \quad 8 \div 4 = 0 \text{ Yes}$$

$$i=5 \quad 8 \div 5 = 0 \text{ NO}$$

$$i=6 \quad 8 \div 6 = 0 \text{ NO}$$

$$i=7 \quad 8 \div 7 = 0 \text{ NO}$$

$$i=8 \quad 8 \div 8 = 0 \text{ Yes}$$

$n=8$  is not prime

Q6) WAP program to print all prime nos from 100 to 200

logic

for ( $n=100$ ;  $n \leq 200$ ;  $n++$ )

{  
 $c=0$ ,  
 for ( $i=1$ ;  $i \leq n$ ;  $i++$ )  
 {  
 if ( $n \% i == 0$ )  
 {  
 c++;  
 }  
 }  
}

Displays value of  $n$   
 if  $n$  is prime

```

    }   c++;
    if(c==2)
        printf("%d \t", n);
}

```

```

#include<stdio.h>
int main()
{
    int n,i,c;
    printf("prime nos from 100 to 200 are\n");
    //logic is checked for all value of n from 100 to 200
    for(n=100;n<=200;n++)
    {

```

//the logic works for every n

```

        c=0;
        for(i=1;i<=n;i++)
        {

```

```

            if(n%i==0)
                c++;
        }

```

//if n is prime display n

```

        if(c==2)
            printf("%d ",n);
    }

```

```

    return 0;
}
```

if n is prime  
display n

repeat it  
for all  
value of n  
from 100 to  
200.

Q7) WAP program to determine a number is  
perfect square or not. (Square root is 0 or > perfect square)

Logic  $\Rightarrow$   $n=16$

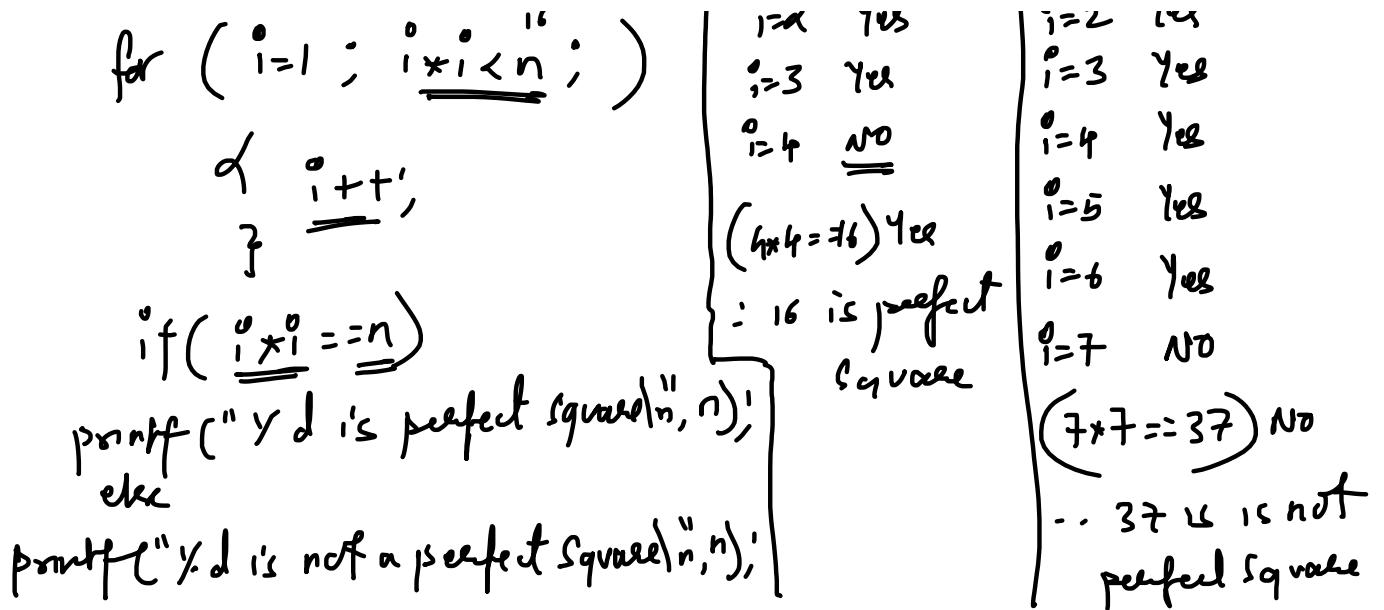
for ( i=1 ;  $i \times i < n$  ; )

$$n=16$$

<u>i=1</u>	Yes
<u>i=2</u>	Yes
<u>i=3</u>	Yes

$$n=37$$

<u>q=1</u>	Yes
<u>q=2</u>	Yes
<u>q=3</u>	Yes



```
#include<stdio.h>
```

```
int main()
{
    int n,i;
    printf("enter the number\n");
    scanf("%d",&n);

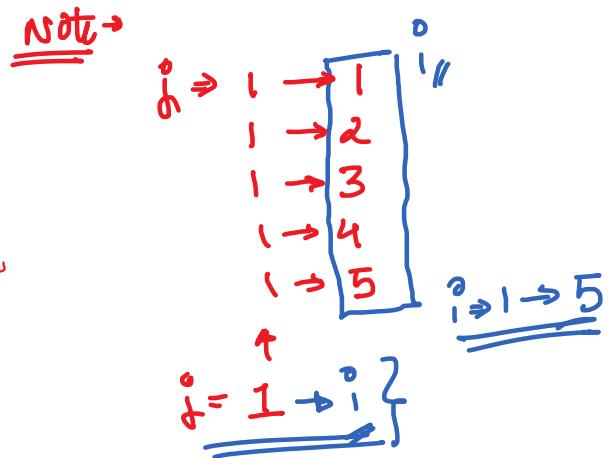
    for(i=1;i*i<n)
    {
        i++;
    }

    if(i*i==n)
        printf("%d is a perfect square\n",n);
    else
        printf("%d is not a perfect square\n",n);
    return 0;
}
```

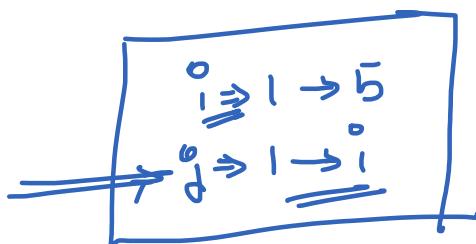
Patterns  $\rightarrow$  10 marks / 80 marks (Online - 1 Q)

		col $\rightarrow$ (j)			
rows		i=1	1	2	3
(i)	i=2	1	2		
	i=3	1	2	3	
	i=4	1	2	3	4
	i=5	1	2	3	4
					5

Let  $i \rightarrow$  iterate through rows  
 $j \rightarrow$  iterate through col.



Note



$\text{for } (i=1; i \leq 5; i++)$

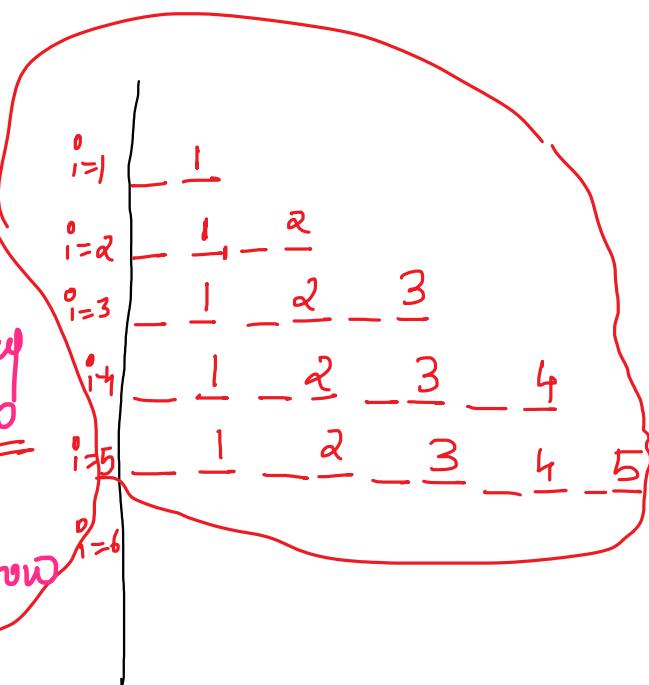
{

$\quad \text{for } (j=1; j \leq i; j++)$

$\quad \quad \text{printf}("%d", j);$

}

$\quad \quad \text{printf}("\n"); \Rightarrow \text{for new row}$



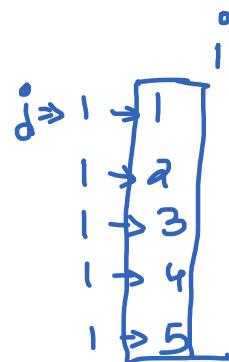
```

#include<stdio.h>
int main()
{
    int i,j;
    for(i=1;i<=5;i++)
    {
        for(j=1;j<=i;j++)
        printf("%2d",j);

        printf("\n");
    }
    return 0;
}

```

$i=1 \rightarrow 1$   
 $i=2 \rightarrow \underline{2} \quad \underline{2}$   
 $i=3 \rightarrow \underline{3} \quad \underline{3} - \underline{3}$   
 $i=4 \rightarrow \underline{\underline{4}} \quad \underline{\underline{4}} \quad \underline{\underline{4}} \quad \underline{\underline{4}}$   
 $i=5 \rightarrow \underline{\underline{\underline{5}}} \quad \underline{\underline{\underline{5}}} \quad \underline{\underline{\underline{5}}} \quad \underline{\underline{\underline{5}}} \quad \underline{\underline{\underline{5}}}$



$$\begin{array}{l}
i \Rightarrow 1 \rightarrow 5 \\
j \Rightarrow 1 \rightarrow i \\
\hline
\end{array}$$

point i

$i=1$	$i=2$	$i=3$	$i=4$	$i=5$
$j=1$	$j=1, 2$	$j=1, 2, 3$	$j=1, 2, 3, 4$	$j=1, 2, 3, 4, 5$
1 time	2 time	3 time	4 time	5 time

```

#include<stdio.h>
int main()
{
    int i,j;
    for(i=1;i<=5;i++)

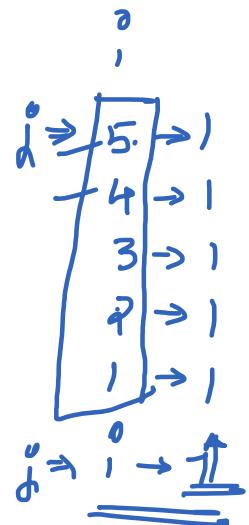
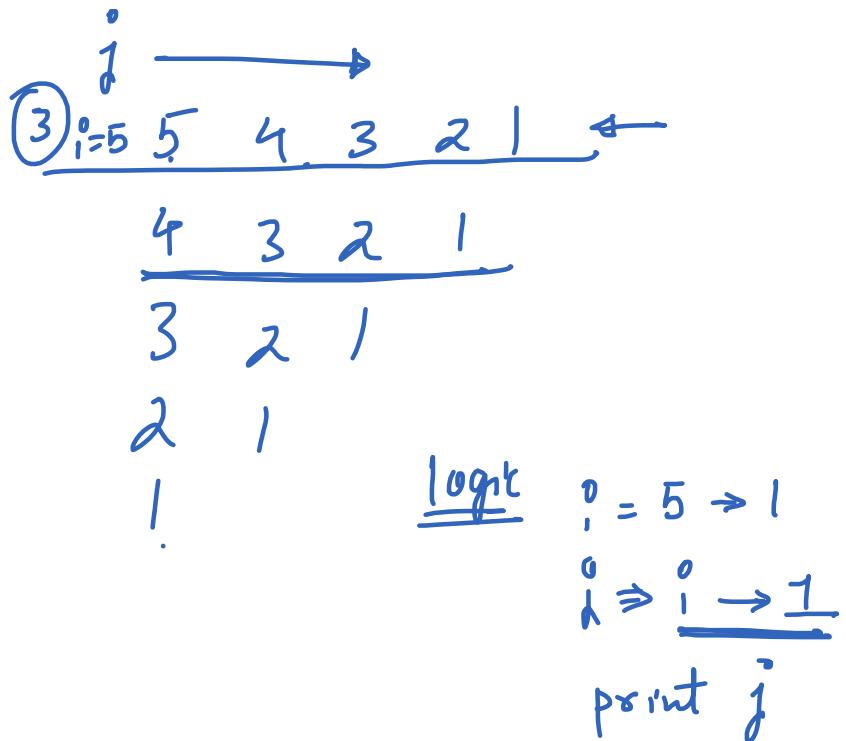
```

```

{
    for(j=1;j<=i;j++)
        printf("%2d",j);

    printf("\n");
}
return 0;
}

```



```

#include<stdio.h>
int main()
{
    int i,j;

    for(i=5;i>=1;i--)
    {
        for(j=i;j>=1;j--)
            printf("%2d",j);

        printf("\n");
    }
    return 0;
}

```

H/W

(1)

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

for wed

(2)

1 a 3 4 5  
2 3 4 5  
3 4 5  
4 5  
5

(3)

A  
A B  
A B C  
A B C D  
A B C D E

Noteformatted Output`int x = 1234;``printf("%d", x);`

1234

`printf("%.5d", x);`

1234

right justified

`printf("%-5d", x);`

1234

`printf("%2d", x);`

1234

`float x = 3.14678;``printf("y.f", x);`

3.146780

`printf("%.10f", x);`

(right justified)

Allocate  
total 10  
spacesout of 10 spaces  
there must be  
3 digits after  
floating point

TTTTT

3.147

Right  
Justified.`printf("%.10f", x);`  
left justified

```
#include<stdio.h>
int main()
{
    int x=123;
    float y=3.14678;
    printf("%d\n",x);
    printf("%5d\n",x);
    printf("%-5d\n",x);
    printf("%f\n",y);
    printf("%10.3f\n",y);
    printf("%-10.3f\n",y);
    return 0;
}
```

Select D:\OODD 2021-22\SPA INFT\SPA PROGRAMS\formattedo  
123  
123  
123  
3.146780  
3.147  
3.147

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

②    1    a    3    4    5  
2    3    b    5  
3    4    c  
4    5  
5

③    ✓  
A  
A    B  
A    B    C  
A    B    C    D  
A    B    C    D    E

$\frac{1}{2} \rightarrow$   
 $1\%_2$   
 $\underline{1}$   
 $\underline{2\%}_2 \quad \underline{1\%}_2$   
 $\underline{\underline{0}} \quad \underline{1}$   
 $\underline{3\%}_2 \quad \underline{2\%}_2 \quad \underline{1\%}_2$   
 $\underline{\underline{1}} \quad \underline{0} \quad \underline{1}$   
 $\underline{4\%}_2 \quad \underline{3\%}_2 \quad \underline{2\%}_2 \quad \underline{1\%}_2$   
 $\underline{\underline{0}} \quad \underline{1} \quad \underline{0} \quad \underline{1}$   
 $\underline{5\%}_2 \quad \underline{4\%}_2 \quad \underline{3\%}_2 \quad \underline{2\%}_2 \quad \underline{1\%}_2$   
 $\underline{\underline{1}} \quad \underline{0} \quad \underline{1} \quad \underline{0} \quad \underline{1}$

$$h = \begin{array}{c|c} \textcircled{i} & \\ \hline 1 & \rightarrow 1 \\ 2 & \rightarrow 1 \\ 3 & \rightarrow 1 \\ 4 & \rightarrow 1 \\ 5 & \rightarrow 1 \end{array}$$

$$\textcircled{j} = \textcircled{i} \rightarrow 1$$

$$\textcircled{j} = \textcircled{i} \rightarrow 1$$

print  $\textcircled{j}\%_2$

!   
 n rows

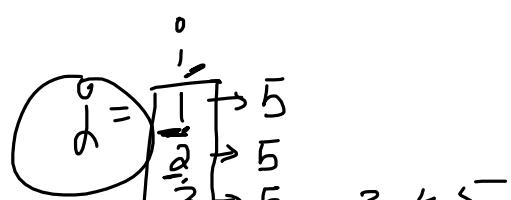
for any  $n$       OR  
 $\frac{n}{a}$       1

```
#include<stdio.h>
int main()
{
    int i,j,n;
    printf(" enter no of rows\n");
    scanf("%d",&n);

    for(i=1;i<=n;i++)
    {
        for(j=i;j>=1;j--)
            printf("%2d",j%2);

        printf("\n");
    }
    return 0;
}
```

(2)  $i = 1 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5$   
 $i = 2 \quad 2 \quad \underline{3 \quad 4} \quad 5$   
 $i = 3 \quad \underline{3 \quad 4 \quad 5}$   
 $i = 4 \quad 4 \quad 5$   
 $i = 5 \quad 5$



$$i = 1 \rightarrow 5$$

$$d = \frac{i}{a} \rightarrow 5$$

print d

```

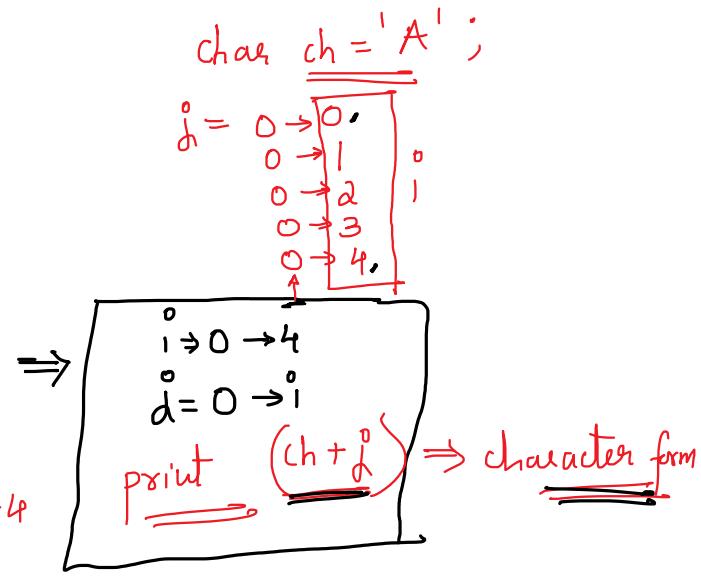
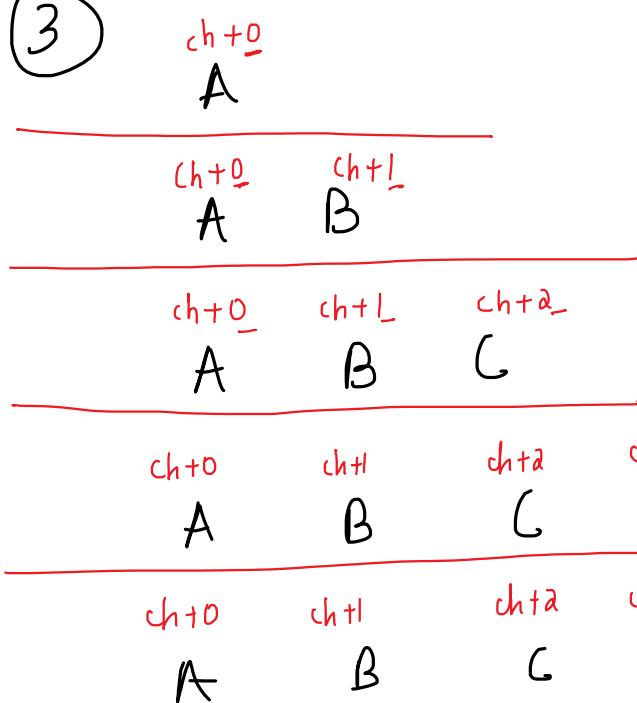
#include<stdio.h>
int main()
{
    int i,j;

    for(i=1;i<=5;i++)
    {
        for(j=i;j<=5;j++)
            printf("%2d",j);

        printf("\n");
    }
    return 0;
}

```

③

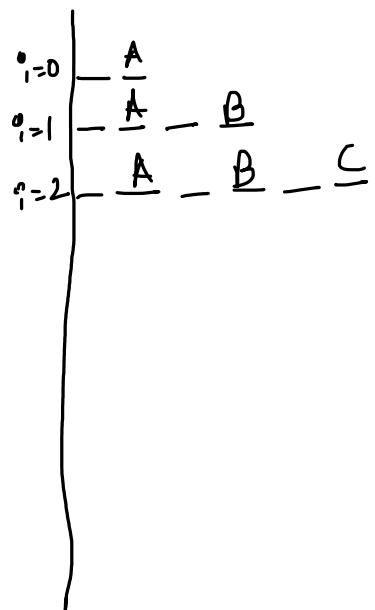


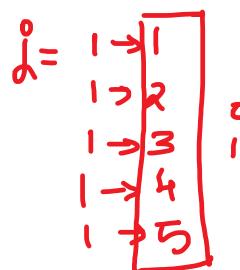
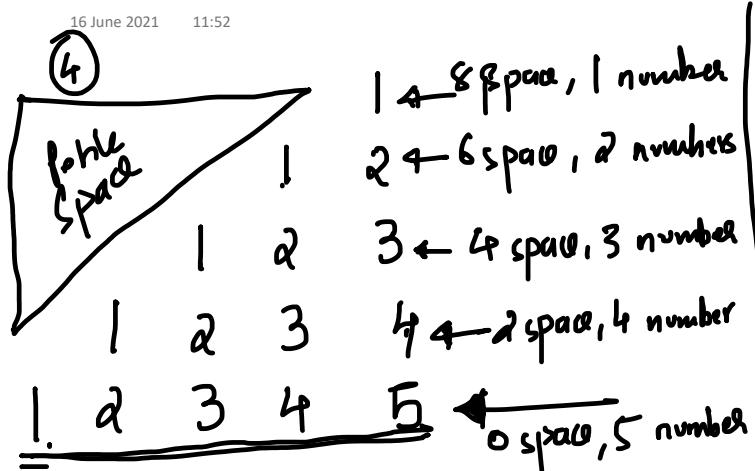
```

#include<stdio.h>
int main()
{
    int i,j;
    char ch='A';
    for(i=0;i<=4;i++)
    {
        for(j=0;j<=i;j++)
            printf("%2c",ch+j);
        printf("\n");
    }
    return 0;
}

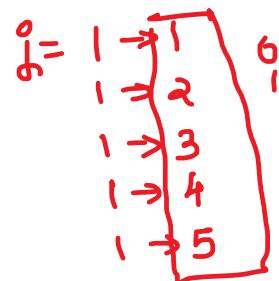
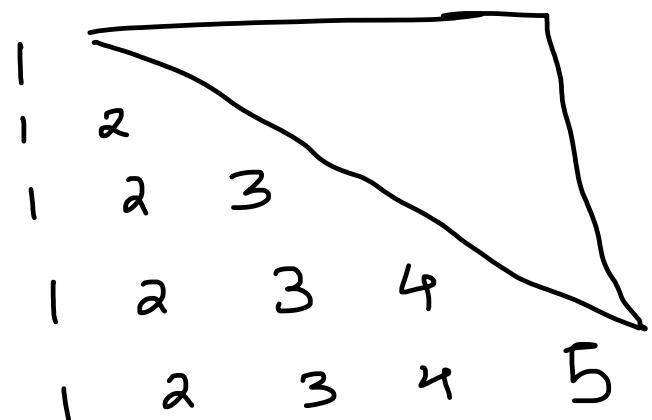
```

$i=0 \quad j=0 \quad ch+0 = A$   
 $i=1 \quad j=0 \quad ch+0 = A$   
 $i=1 \quad j=1 \quad ch+1 = B$   
 $i=2 \quad j=0 \quad ch+0 = A$   
 $i=2 \quad j=1 \quad ch+1 = B$   
 $i=2 \quad j=2 \quad ch+2 = C$





```
i = 1 → 5
j = 1 → i
print j
```



```
i = 1 → 5
j = 1 → i
print j
```

↑ Yahan  
pehle  
j print hota  
hui

Yahan  $i$  in every row

pehle Kuch spaces print  
huge then  $j$  print hoga-

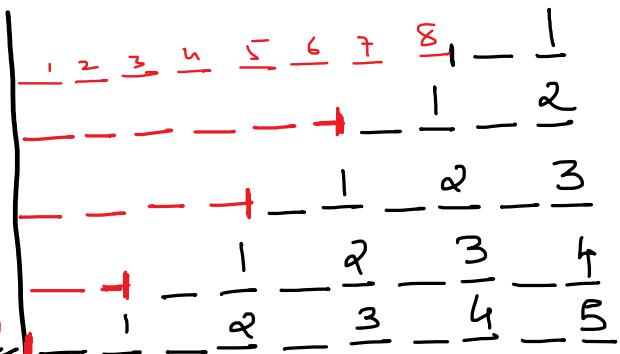


how to do

(we will create a Space  
Wala loop)

. r  $\rightarrow$  0 to

$S=8$   
 $S=6$   
 $S=4$   
 $S=2$   
 $S=0$



Wuaoiu | |  $\xrightarrow{s=0}$  | - - -  $\alpha$  - = - - - =  
Let  $s$  be no of spaces to

NOW       $\text{int } i, j, S = 8; \text{int } K;$

// for now

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

// In every row PointSpace first

```
for( k=1 ; k<=s ; k++ )
```

printf(" "); ← Single Space 5 Times

Space  
wala loop -

// Print j value

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

```
printf("%d", j);
```

s = s - a; = pointf("ln");

```
#include<stdio.h>
```

```
int main()
```

1

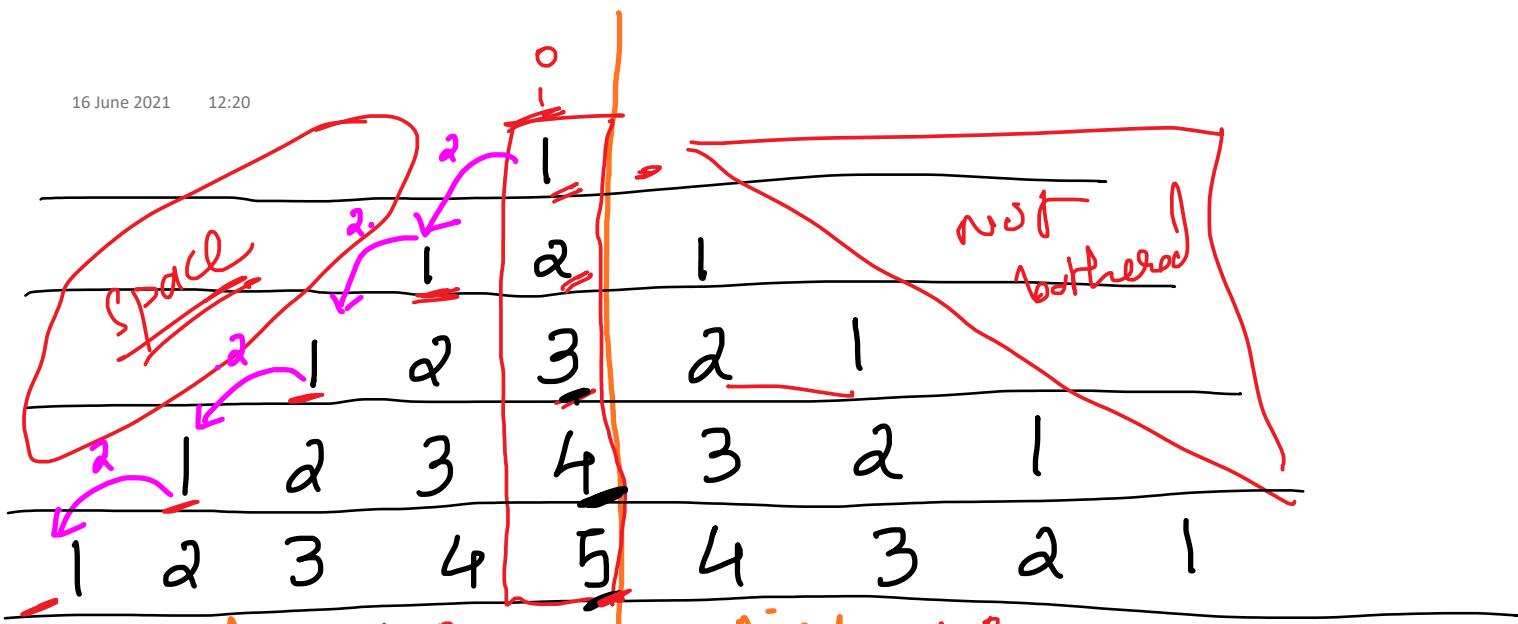
```
int i,j,k,s;
```

s=8;

```
for(i=1;i<=5;i++)
{
    //space wala loop
    for(k=1;k<=s;k++)
        printf(" ");

    //print j value
    for(j=1;j<=i;j++)
        printf("%2d",j);

    s=s-2;
    printf("\n");
}
return 0;
}
```

left side  $j$ 

$$\begin{array}{c} j = 1 \rightarrow 1 \\ \downarrow \\ j = 2 \rightarrow 2 \\ \downarrow \\ j = 3 \rightarrow 3 \\ \downarrow \\ j = 4 \rightarrow 4 \\ \downarrow \\ j = 5 \rightarrow 5 \end{array}$$

$$j \Rightarrow 1 \rightarrow 5$$

$$j \Rightarrow 1 \rightarrow j$$

$$\boxed{S = 8}$$

for ( $i=1$  ;  $i \leq 5$  ;  $i++$ )

right side  $j$ 

$$j = i-1 \text{ to } 1$$

// space loop

```
for(k=1; k <= S; k++)
    printf(" ");
```

// left wala

```
for(j=1; j <= i; j++)
    printf("%d", j)
```



// Right wala

for(j=i-1; j>=1; j--)

printf("%2d", j);

s=s-2;

printf("\n"); ← next row

```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
    int i,j,k,s;
```

```
    s=8;
```

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

```
{
```

```
        //space wala loop
```

```
        for(k=1;k<=s;k++)
```

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

```
        //print left j value
```

```
        for(j=1;j<=i;j++)
```

```
            printf("%2d",j);
```

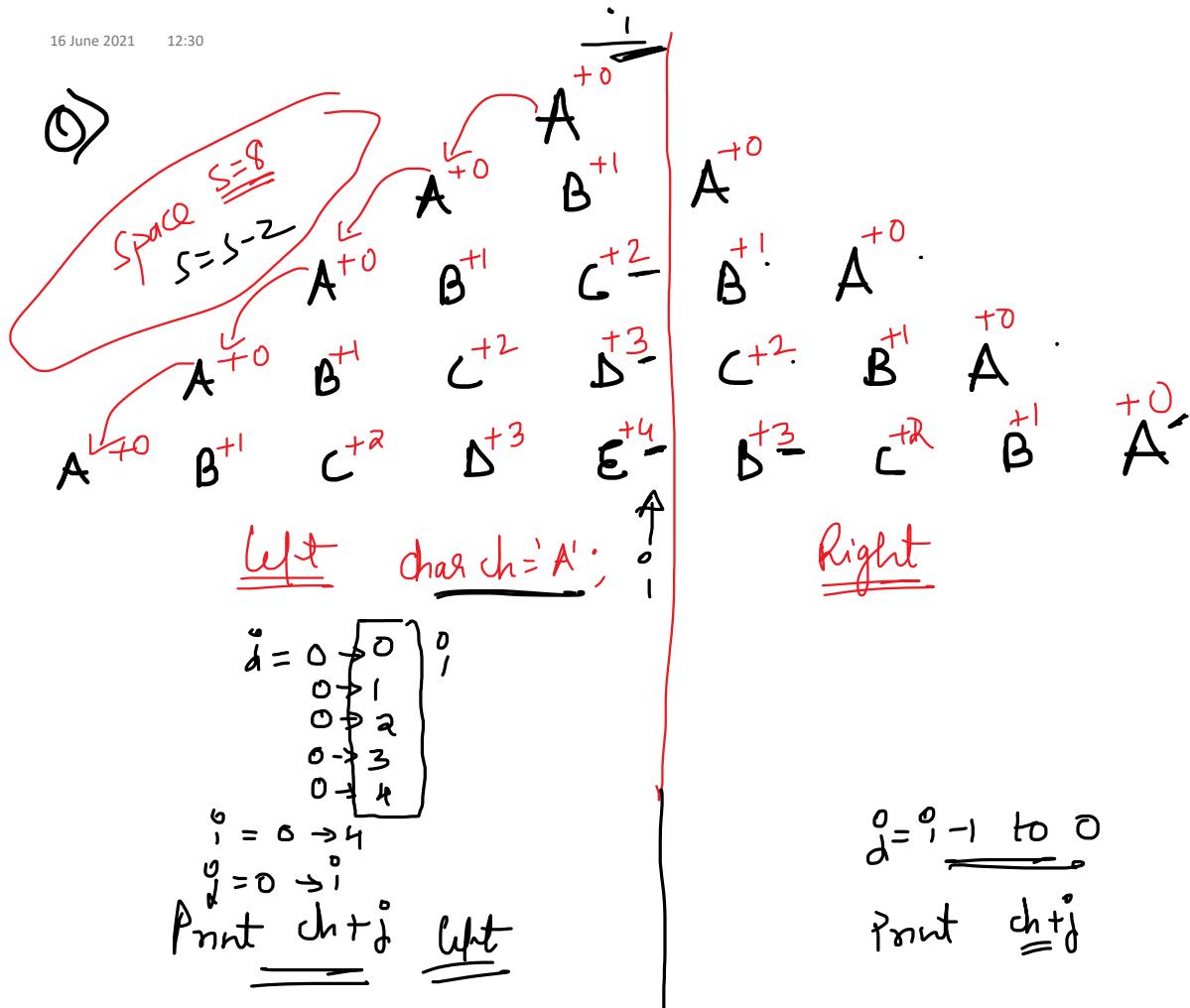
```
        //print right j value
```

```
        for(j=i-1;j>=1;j--)
```

```
printf("%2d",j);

s=s-2;
printf("\n");
}

return 0;
}
```



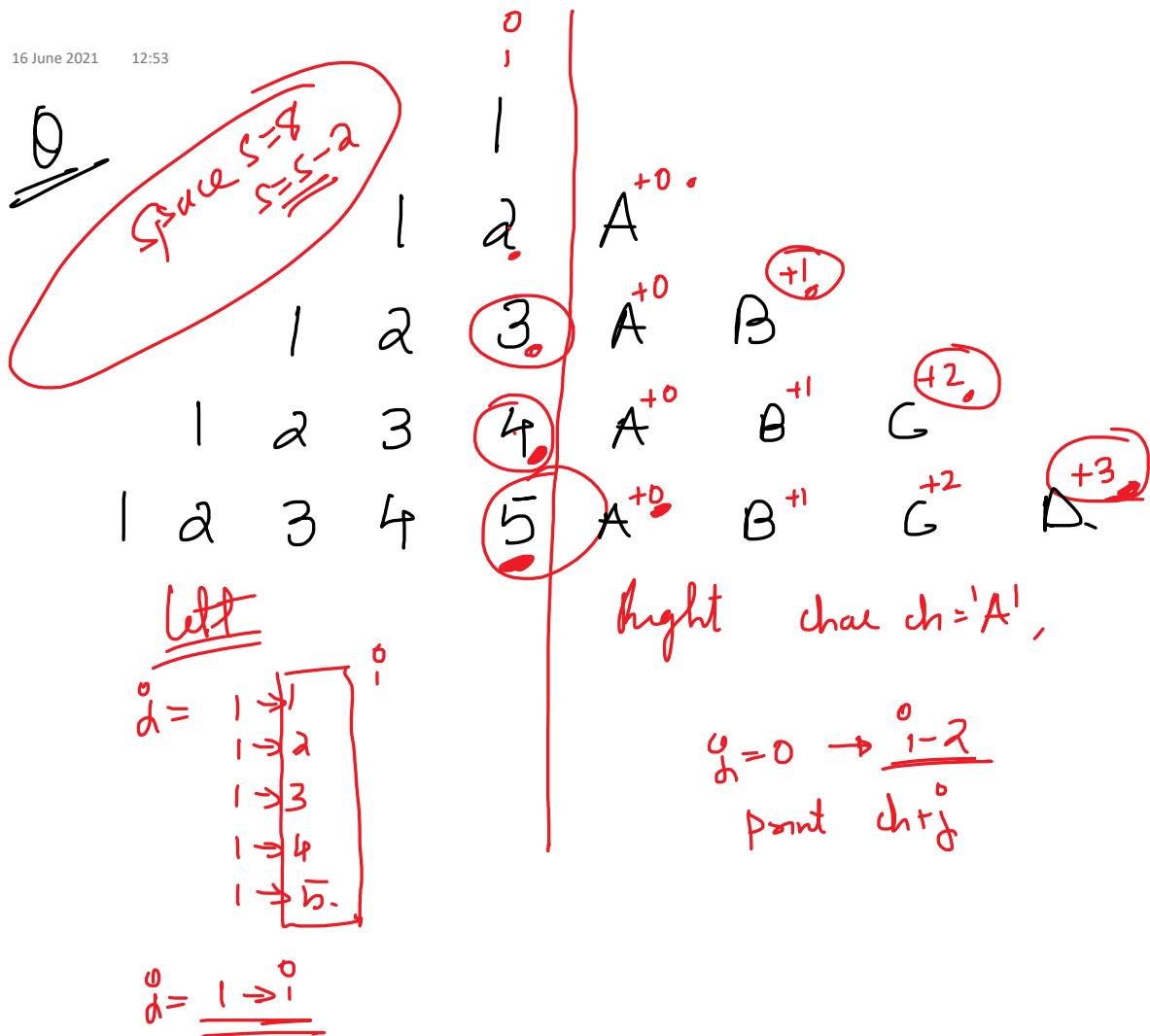
```
#include<stdio.h>
int main()
{
    int i,j,k,s,n;
    char ch='A';

    s=8;
    for(i=0;i<=4;i++)
    {
        //space wala loop
        for(k=1;k<=s;k++)
            printf(" ");

        //print left j value
        for(j=0;j<=i;j++)
            printf("%2c",ch+j);

        //print right j value
    }
}
```

```
for(j=i-1;j>=0;j--)  
printf("%2c",ch+j);  
  
s=s-2;  
printf("\n");  
}  
return 0;  
}
```



```
#include<stdio.h>
int main()
{
    int i,j,k,s,n;
    char ch='A';

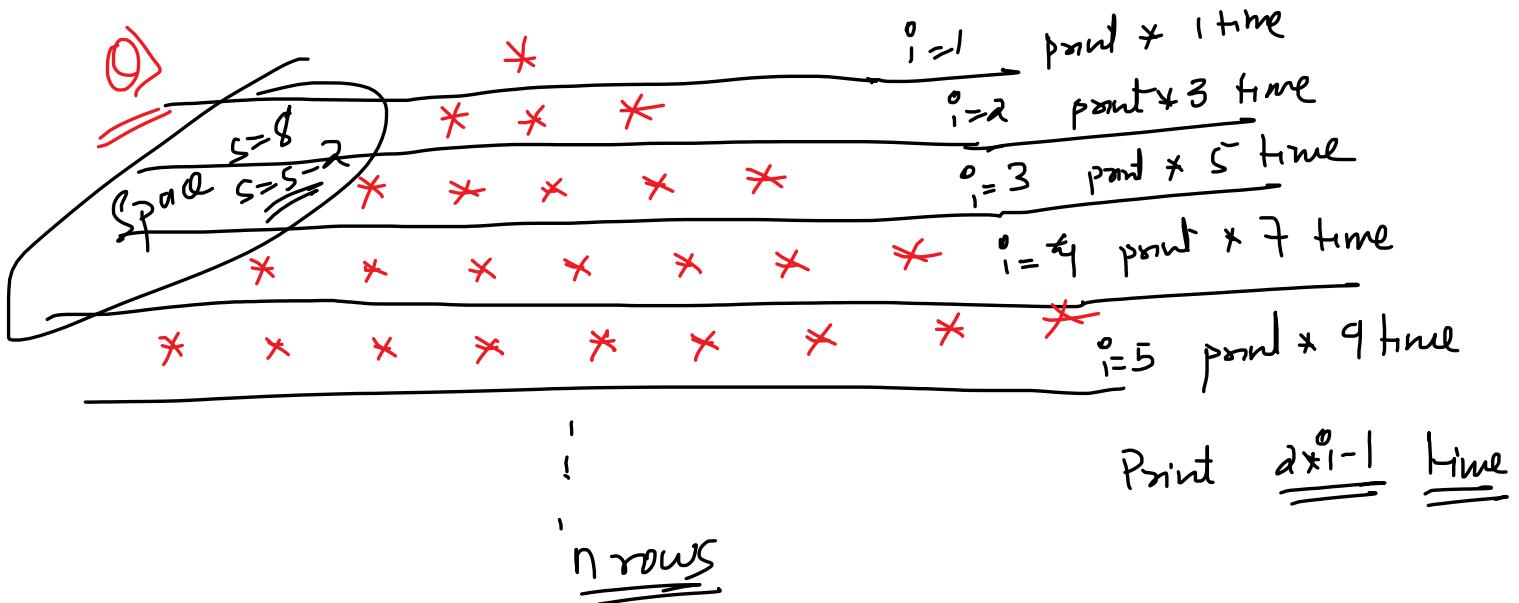
    s=8;
    for(i=1;i<=5;i++)
    {
        //space wala loop
        for(k=1;k<=s;k++)
            printf(" ");

        //print left j value
        for(j=1;j<=i;j++)
            printf("%2d",j);
    }
}
```

```
//print right j value
for(j=0;j<=i-2;j++)
printf("%2c",ch+j);

s=s-2;
printf("\n");
}

return 0;
}
```



```
#include<stdio.h>
int main()
{
    int i,j,k,s,n;
    char ch='*';

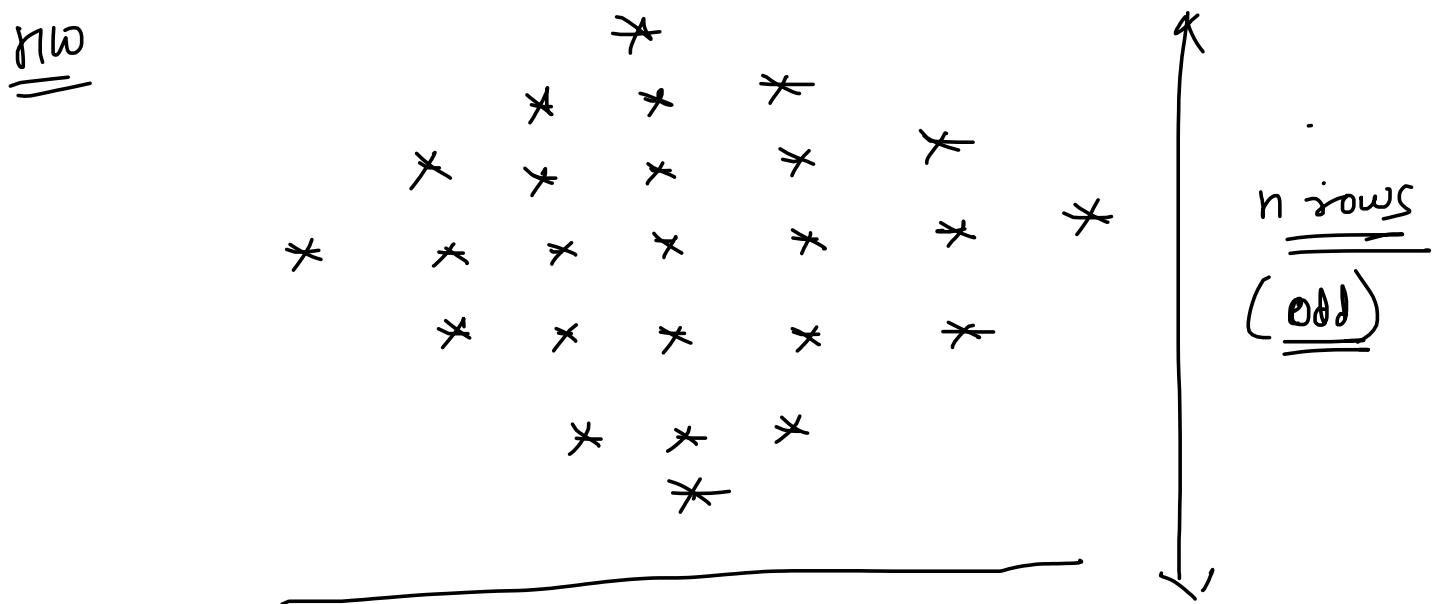
    printf(" enter no of rows\n");
    scanf("%d",&n);

    s=2*(n-1);
    for(i=1;i<=n;i++)
    {
        //space wala loop
        for(k=1;k<=s;k++)
            printf(" ");

        //print *
        for(j=1;j<=2*i-1;j++)
            printf("%2c",ch);

        s=s-2;
    }
}
```

```
    printf("\n");
}
return 0;
}
```



Expl ⇒ WACP to calculate Simple & Compound Interest

formula  $SI \Rightarrow \frac{P \times n \times r}{100}$

$$CI = P \times \left( \left( 1 + \frac{r}{100} \right)^n - 1 \right)$$

$$pow(x, y) \Rightarrow x^y$$

Read float  $P, n, r, SI$ ;  
 $\text{scanf}("if you f.y.f", &P, &n, &r);$

$$SI = (P \times n \times r) / 100;$$

print  $SI$

$$CI = \frac{P * pow\left(1 + \frac{r}{100}, n\right) - P}{100}$$

print  $CI$

//Expt : 1 Simple and Compound interest

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
int main()
{
    float p,r,s,n,c;

    printf("Enter principle amount, number of years and ROI:");
    scanf("%f %f %f", &p, &n, &r);
    s = (p*n*r)/100;
    c = p*pow(1+r/100, n)-p;      //r/100 will be solved first
    printf("Simple interest = %f", s);
    printf("\nCompound interest = %f", c);
    return 0;
}
```

Expt: WAP to read two points x coord, y coord from user & calculate distance bet'n these two points

logic

float  $x_1, y_1, x_2, y_2$

dist  $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

```
#include<stdio.h>
#include<math.h>
int main()
{
    float x1,y1,x2,y2;
    float dist;

    printf("Enter x,y coordinates of first point:");
    scanf("%f%f",&x1,&y1);

    printf("Enter x,y coordinates of second point:");
    scanf("%f%f",&x2,&y2);

    dist = sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1));
    printf("The distance between points = %f",dist);

    return 0;
}
```



$$\text{dist} = \sqrt{\left( \underbrace{\text{pow}(x_1 - x_2, 2)}_{a} + \underbrace{\text{pow}(y_1 - y_2, 2)}_{b} \right)};$$
$$\text{dist} = \sqrt{(x_1 - x_2) * (x_1 - x_2) + (y_1 - y_2) * (y_1 - y_2)};$$
$$\left\{ \begin{array}{l} a = \text{pow}(x_1 - x_2, 2); \\ b = \text{pow}(y_1 - y_2, 2); \end{array} \right| \left\{ \begin{array}{l} a = (x_1 - x_2) * (x_1 - x_2); \\ b = (y_1 - y_2) * (y_1 - y_2); \end{array} \right.$$
$$\text{dist} = \sqrt{a+b};$$

③ WACP to chk constructability of a triangle  
Using 3 values (Expected 3 sides) and if Yes then  
Specify type

Logic let float a, b, c;

(Equilateral / Isosceles)

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

// Sum of any 2 side must be greater than 3<sup>rd</sup> side

if ((a+b>c) && (b+c>a) && (a+c>b))

P      printf(" The values forms sides of Triangle\n");

    if ((a==b) && (b==c))

        printf(" Equilateral triangle\n");

    else if ((a==b) || (b==c) || (c==a))

        printf(" Isosceles triangle\n");

    else      printf(" Scalene Triangle\n");

    else

        printf(" The values does not form sides of triangle\n");

//Expt : 3 Constructability of Triangle

```
#include<stdio.h>
int main()
{
    float a,b,c;
    printf("Enter three values:");
    scanf("%f %f %f",&a,&b,&c);

    if(((a+b)>c)&&((b+c)>a)&&((a+c)>b))
    {
        printf("the values forms sides of traingle\n");
        if((a==b)&&(b==c))
            printf("Equilateral triangle");

        else if((a==b) || (b==c) || (a==c))
            printf("Isosceles triangle");

        else
            printf("Scalene triangle");
    }
    else
        printf("The values do not form sides of triangle\n");

    return 0;
}
```

## Q4) Roots of Quadratic Equation

```
#include<stdio.h>
#include<math.h>
int main()
{
    float a,b,c,x1,x2,d; //declare every variables at the start
    printf("Enter the coefficients of quadratic equation\n");
    scanf("%f %f %f",&a,&b,&c);

    if(a==0)
    {
        printf("entered coefficients does not form quadratic equation\n");
    }
    else
    {
        printf("entered coefficients forms quadratic equation\n");
        d=(b*b)-(4*a*c);

        if(d>0)
        {
            printf("roots are real and distinct\n");
            x1=(-b+sqrt(d))/(2*a);
            x2=(-b-sqrt(d))/(2*a);
            printf("Roots of the quadratic eqns are %f and %f",x1,x2);
        }

        else if(d==0)
    }
}
```

```
{  
printf("roots are real and equal\n");  
x1=-b/(2*a);  
x2=x1;  
printf("Roots of the quadratic eqns are %f and %f",x1,x2);  
}  
else  
printf("roots are imaginary\n");  
}//outer else  
  
return 0;  
}//MAIN
```

⑤ Arithmetic Calculator } Use Switch Case

<u>Input</u>	<u>char ch;</u>	'+'	for addition
		' - '	for Subtraction
		' * '	for Multiplication
		' / '	for Division

Input two float values  $\rightarrow$  float a, b;  
float result;

```
printf("Enter choice\n");
scanf("%c", &ch);
```

```
printf("Enter two numbers\n"),
scanf("%f %f", &a, &b);
```

switch(ch)

```
{ case '+': result = a+b;
    printf("Addition = %f\n", result),
    break;
```

case '-':

```

        case '*':
        case '/':
    }
    default: printf("Invalid choice\n");

```

```

//Expt 5:Arithmetic calculator
#include<stdio.h>
#include<math.h>
int main()
{
    char choice;
    float a,b,result;

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

    printf("+ : Addition\n- : Subtraction\n");
    printf("* : Multiplication\n/ : Division");

    fflush(stdin); //flush inputstream before reading any character

    printf("\nEnter your choice:");
    scanf("%c",&choice);

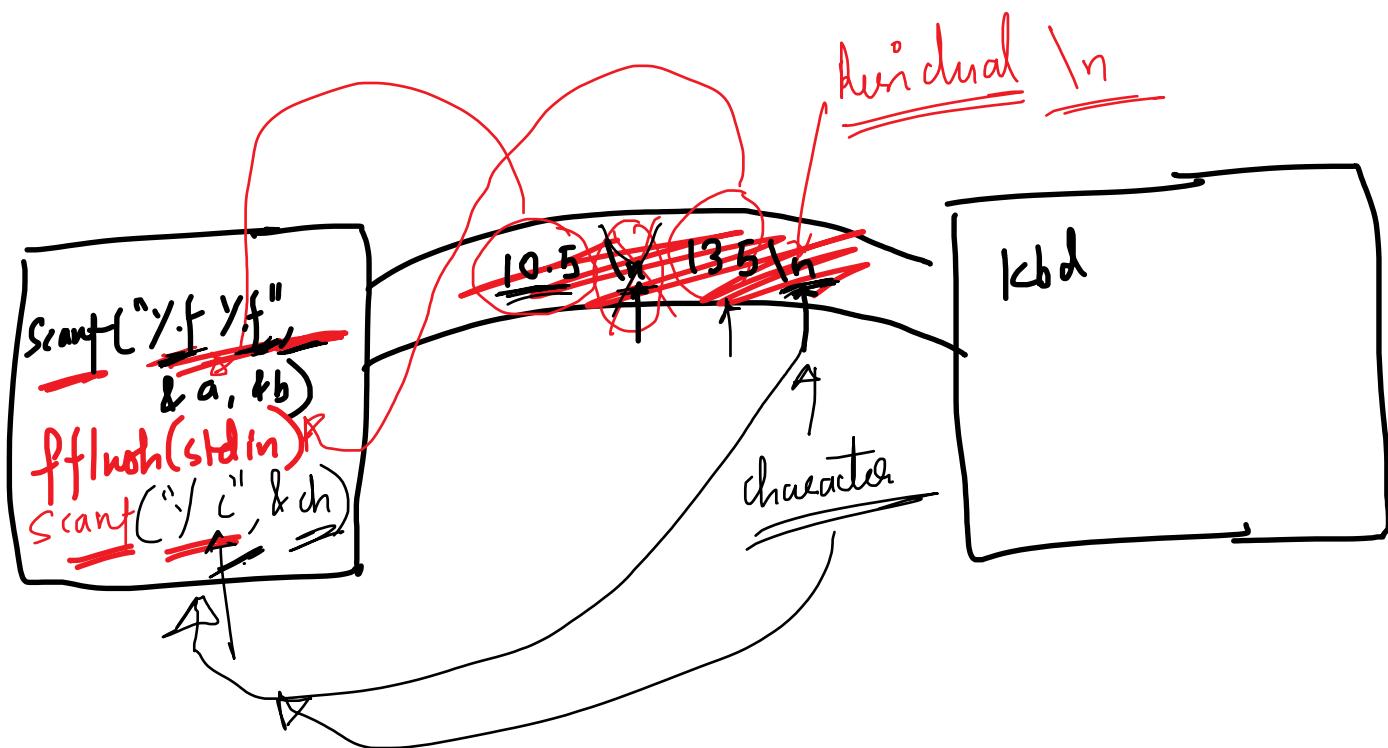
    switch(choice)
    {
        case '+': result = a + b;
                    printf("Addition = %f",result);
                    break;
        case '-': result = a - b;
                    printf("Subtraction = %f",result);
    }
}

```

```

        break;
case '*': result = a * b;
    printf("Product = %f",result);
    break;
case '/': result = a / b;
    printf("Division = %f",result);
    break;
default : printf("Invalid option");
}
return 0;
}

```



Note → When to use fflush:

① If we want to read a character "%c"

if the scanf("%c") is the first scanf() of the prog then no need to use fflush

② If scanf("%c") is ~~is~~ not the first scanf() of the program then before scanf("%c") we must specify

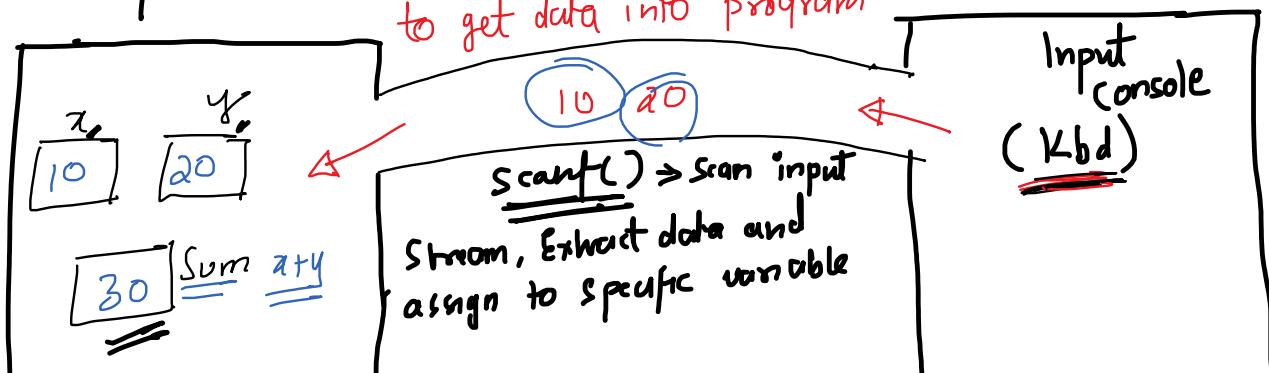
fflush(stdin);

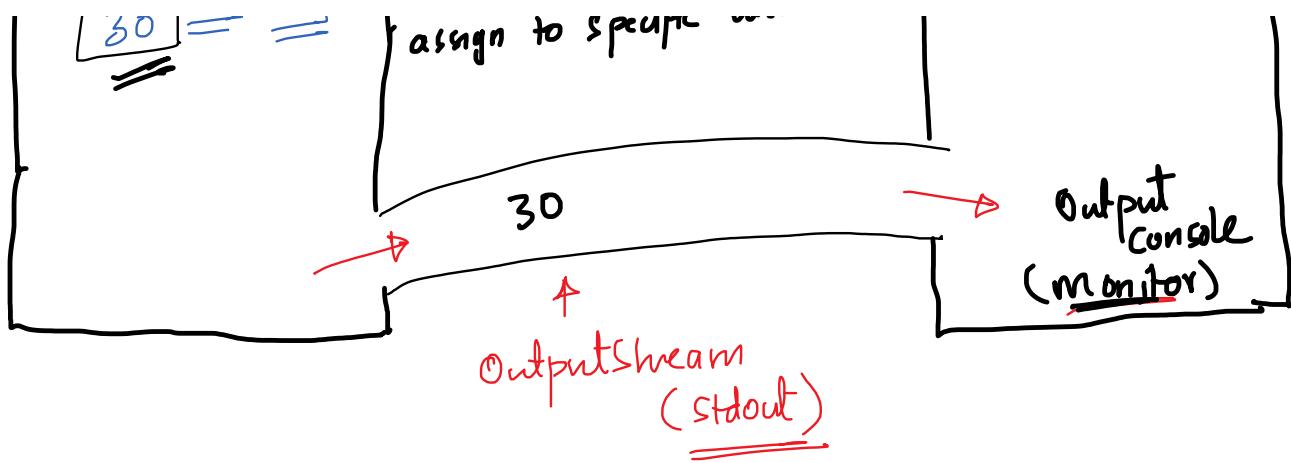
stdin → Standard Input Stream

Program

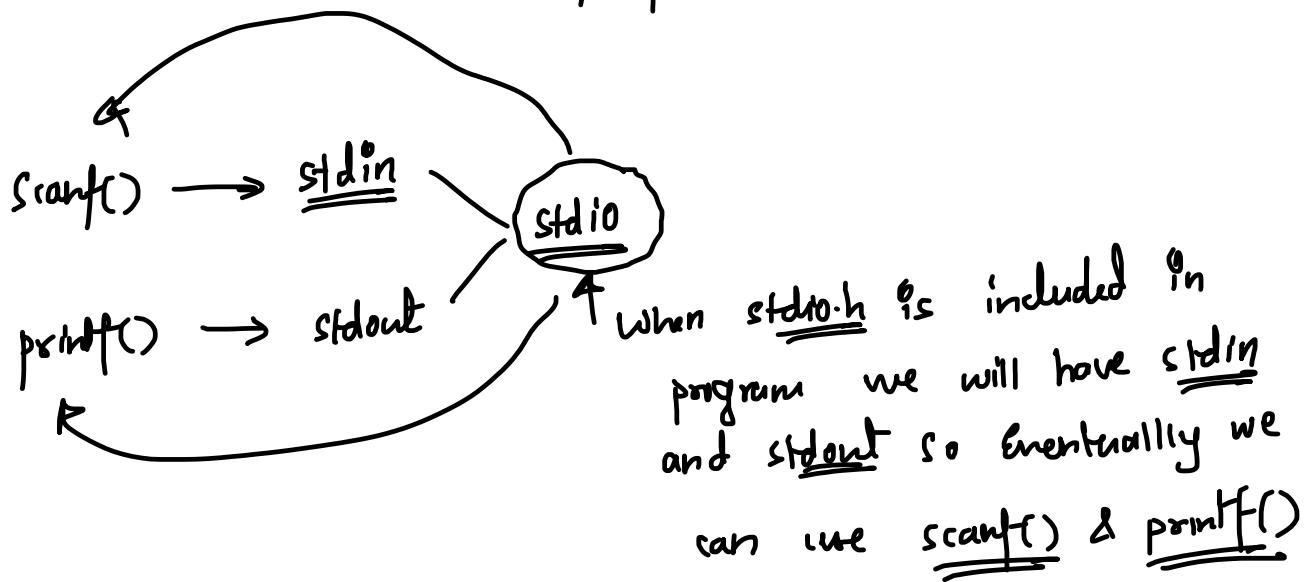
InputStream (stdin)  
to get data into program

I/O console





`point( )` → It takes the data from program,  
 places in OutputStream and then  
 display on Monitor



## ② while loop

Syntax → `while ( condition )`

of  
body of while loop  
}

1. It is Entry controlled loop

2 If the condition is true then only entry is allowed in the loop

Consider (Using for loop)

```
int i;
for(i=1; i<=10; i++)
    printf("Hello\n");
```

operator =

Using while loop.

```
int i=1; // initialis
while(i<=10) // condition
{
    printf("Hello\n");
    i++; // Exp
}
```

operator =

Ideally →  
We will prefer while loop when the operator is initialized at run time

③ WAP Program to Count no of digits in a given Integer

Logic      Run Time  
 $n = 1234$ ,  $c = 0$

Note       $n = 123$

Logic       $n = \underline{\underline{1234}}, c = \underline{\underline{0}}$

$$\left\{ \begin{array}{l} n = \underline{\underline{n/10}} \Rightarrow 1234/10 \Rightarrow \underline{\underline{123}} \\ c = c++ \Rightarrow 1 \end{array} \right.$$

$$\left\{ \begin{array}{l} n = \underline{\underline{n/10}} \Rightarrow 123/10 \Rightarrow \underline{\underline{12}} \\ c++ ; \Rightarrow 2 \end{array} \right.$$

$$\left\{ \begin{array}{l} n = \underline{\underline{n/10}} \Rightarrow 12/10 \Rightarrow \underline{\underline{1}} \\ c++ \Rightarrow 3 \end{array} \right.$$

$$\left\{ \begin{array}{l} n = \underline{\underline{n/10}} \Rightarrow 1/10 \Rightarrow \underline{\underline{0}} \\ c++ \Rightarrow \underline{\underline{4}} \text{ Ans} \end{array} \right.$$

Do  $\underline{\underline{0}} \neq 0$  take  $\underline{\underline{n/10}}$

Note       $\underline{\underline{n=123}}$   
 $10 \overline{) \underline{\underline{123}}} \rightarrow \text{quotient}$   
 $\underline{\underline{12}} \overline{) \underline{\underline{23}}} \rightarrow \text{remainder}$   
 $\underline{\underline{23}} \overline{) \underline{\underline{20}}} \rightarrow \underline{\underline{3}}$   
 $\underline{\underline{20}} \overline{) \underline{\underline{3}}} \rightarrow \underline{\underline{0}}$

$$\underline{\underline{n/10}} \Rightarrow \underline{\underline{12}}$$

$$\underline{\underline{n \% 10}} \Rightarrow \underline{\underline{3}}$$

$\underline{\underline{n \% 10}}$  gives me last digit  
of number

$\underline{\underline{n/10}}$  give me a number  
excluding last digit

Logic  $\Rightarrow$  Using for loop

int i, n, c = 0;

printf("Enter the number\n");  
scanf("%d", &n); // Initialize

for( ;  $\underline{\underline{n \neq 0}} ; n = \underline{\underline{n/10}}$  )  
c++;

printf("No of digits = %d\n", c);

Using while loop.

int i, n, c = 0;

printf("Enter the number\n");

scanf("%d", &n); // Initialize

while( $\underline{\underline{n \neq 0}}$ )

{ c++; }

    n =  $\underline{\underline{n/10}}$ ,

}

printf("No of digits is %d\n", c);

Preferred.

As the iterator n is initialized

for  
Yahan Se Yahan Tak

As the iterator n is initialized  
at Run Time  
while  $\Rightarrow$  jab tak

```
#include<stdio.h>
int main()
{
    int n,n1,c=0;

    printf("enter the number\n");
    scanf("%d",&n);
    n1=n;
    while(n1!=0)
    {
        n1=n1/10;
        c++;
    }
    printf("nos of digits in %d is %d\n",n,c);
    return 0;
}
```

Q) Read a Number from User & Determine it is Armstrong Number or not?

Armstrong nos  $\Rightarrow$  if Sum of digits of the number raised to the power of total no of digits equals to the original number itself then the number is Armstrong Number

Ex  $n = 153$

Count no of digits = 3

No  $1^3 + 5^3 + 3^3 \Rightarrow 1 + 125 + 27 \Rightarrow 153$

153 is Armstrong no

Ex  $n = 1634$

Total No of digits = 4  $\Rightarrow$  No  $1^4 + 6^4 + 3^4 + 4^4 \Rightarrow 1 + 1296 + 81 + 256 \Rightarrow 1634$

Ex  $n = 8208$

No of digits = 4

No. of digits = 4

Now ⇒

$$4 + 4 + 4 + 4$$

$$= 4096 + 16 + 0 + 4096$$

$$\Rightarrow 8208$$

Armstrong No

Logic → Let n be the Number

Print n, s=0, Y, n1; c=0,

printf("Enter the number \n");  
scanf("%d", &n);

n1=n;

// Step 1: Count no of digits

```
while(n1 != 0)  
{  
    c++;  
    n1 = n1 / 10;  
}
```

// c → gives total no of digits

Let  
n1 = 1634, s=0, c=0

n1 = 1634

// count

c=1	c=2	c=3	c=4
n1 = 163	n1 = 16	n1 = 1	n1 = 0

c=4

Step 2 → Extract each digit and raised to power of c  
and add it to the sum

n1=n;

while(n1 != 0)

    Y = n1 % 10;

    S = S + pow(Y, c);

    n1 = n1 / 10;

s=0

n1 = 1634, c=4

Y = 1634 % 10 ⇒ 4

S = 0 + pow(4, 4)

S ⇒ 256

n1 = 1634 / 10 ⇒ 163

(2)

1

2

3

4

5

6

7

8

9

0

Y = 163 / 10 ⇒ 3

S = 256 + pow(3, 4)

= 256 + 81 ⇒ 337

n1 = 16

(3)

$s = s + \underline{\underline{\text{pow}(r, c)}}$   
 $n1 = n1 / 10;$   
 if ( $s == n$ )  
 printf("y.d is Armstrong \n", n);  
 else  
 printf("y.d is Not Armstrong \n", n);

$\textcircled{3}$ $r = 16 \% 10 = 6$ $s = 337 + \text{pow}(6, 4)$ $= 337 + 1296$ $= 1633$ $n1 = 16 / 10 = \underline{\underline{1}}$	$y = 1 \% 10 = 1$ $s = 1633 + \text{pow}(1, 4)$ $= 1633 + 1$ $= \underline{\underline{1634}}$ $n1 = \underline{\underline{0}}$
---	--

Since  $s = 1634$   
 $n = 1634$

$s == n \Rightarrow \text{Armstrong}$

Imp

```

#include<stdio.h>
#include<math.h>
int main()
{
  int n,s,n1,r,c;

  printf("enter the number\n");
  scanf("%d",&n);
  //count no of digits
  n1=n;
  c=0;
  while(n1!=0)
  {
    n1=n1/10;
    c++;
  }
  // c gives no of digits
  //logic of armstrong
  n1=n;
  s=0;
  while(n1!=0)
  {
    r=n1%10;
    s=s+pow(r,c);
    n1=n1/10;
  }

  if(n==s)
  printf("number %d is an armstrong number\n",n);
  else

```

```
printf("number %d is not an armstrong number\n",n);
      return 0;

}
```

Q) WAP program to display All Armstrong Nos from 1 to 10000.

for( n=1 ; n<=10000 ; n++ )

//count no of digits

n1=n;

c=0;

while(n1!=0)

{

n1=n1/10;

c++;

}

// c gives no of digits

//logic of armstrong

n1=n;

s=0;

while(n1!=0)

{

r=n1%10;

s=s+pow(r,c);

n1=n1/10;

}

if(n==s)

→ if

→ n==s

Means number is Armstrong

printf("%d\t",n);

→ point that number

}

//for loop

Program →

```
#include<stdio.h>
#include<math.h>
```

```
int main()
{
    int n,s,n1,r,c;
    printf("Armstrong nos from 1 to 10000 are\n");

    for(n=1;n<10000;n++)
    {

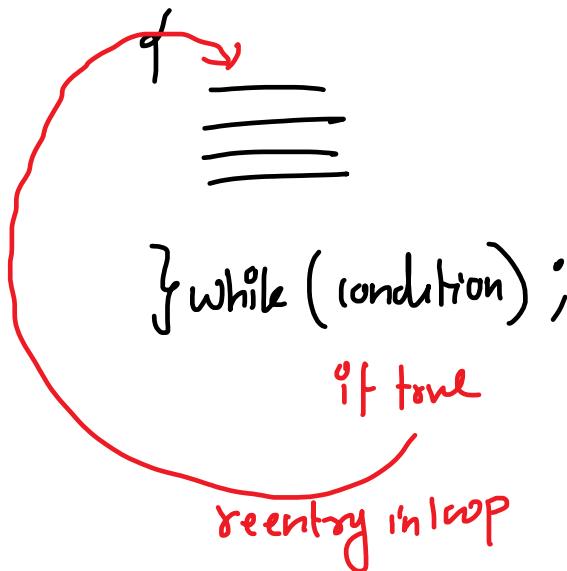
        //count no of digits
        n1=n;
        c=0;
        while(n1!=0)
        {
            n1=n1/10;
            c++;
        }

        //logic of armstrong
        n1=n;
        s=0;
        while(n1!=0)
        {
            r=n1%10;
            s=s+pow(r,c);
            n1=n1/10;
        }

        if(n==s)
        printf("%d\t",n);//display n if it is armstrong
    }
    return 0;
}
```

## do-while loop \*

Syntax , do



Ex

```

int i=11;
do
{
    printf("Hello\n");
} while (i<=10); →
  
```

Op

Hello

```

int i=11;
while(i<=10)
{
    printf("Hello\n");
}
  
```

Op == No output

\* do while is Exit Controlled loop

↳ Here condition is checked at the end of loop

\* if condition is true, then reentry in the

- loop is allowed
- \* if condition is false then reentry in the loop is not allowed
  - \* There is assurance that loop will execute atleast once
  - \* Exit controlled loop (do-while) is used for
- Working    Menu    Driven    Program

```
//Arithmetic calculator Menu Driven
#include<stdio.h>
#include<math.h>
int main()
{
    char choice;
    float a,b,result;

    do
    {

        printf("+ : Addition\n- : Subtraction\n");
        printf("* : Multiplication\n/ : Division\n");
        printf("e/E: Exit\n");

        fflush(stdin); //flush inputstream before reading any character

        printf("\nEnter your choice:\n");
        scanf("%c",&choice);

        printf("Enter two numbers:\n");
        scanf("%f %f",&a,&b);
    }
}
```

```
switch(choice)
{
    case '+': result = a + b;
        printf("Addition = %f\n",result);
        break;
    case '-': result = a - b;
        printf("Subtraction = %f\n",result);
        break;
    case '*': result = a * b;
        printf("Product = %f\n",result);
        break;
    case '/': result = a / b;
        printf("Division = %f\n",result);
        break;

    case 'e':
    case 'E':printf(" Exit Condition\n"); break;

    default : printf("Invalid option\n");
}
}while(choice!='e' && choice!='E');
return 0;
}
```

Q) Imp

WAP program to create a menu where angle is read from user & we have options to display Sine, cosine, tan, Natural log and Log to base 10 of the value

```
#include<stdio.h>
#include<math.h>
int main()
{
    int ch;
    float v,r;

    do
    {
        printf("\n1:sine\n2:cosine\n");
        printf("3:tan\n4:Natural log\n");
        printf("5:Log to base 10\n");
        printf("6:exit\n");

        printf("enter choice\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:printf("enter the angle\n");
                      scanf("%f",&v);
                      r=(v*3.14)/180;// converted to radian
                      printf("sin= %f\n",sin(r));break;

            case 2:printf("enter the angle\n");
                      scanf("%f",&v);
                      r=(v*3.14)/180;
                      printf("cos= %f\n",cos(r));break;

            case 3:printf("enter the angle\n");
```

```
scanf("%f",&v);
r=(v*3.14)/180;
printf("tan= %f\n",tan(r));break;

case 4:printf("enter the value\n");
scanf("%f",&v);
printf("log= %f\n",log(v));break;

case 5:printf("enter the value\n");
scanf("%f",&v);
printf("log10= %f\n",log10(v));break;

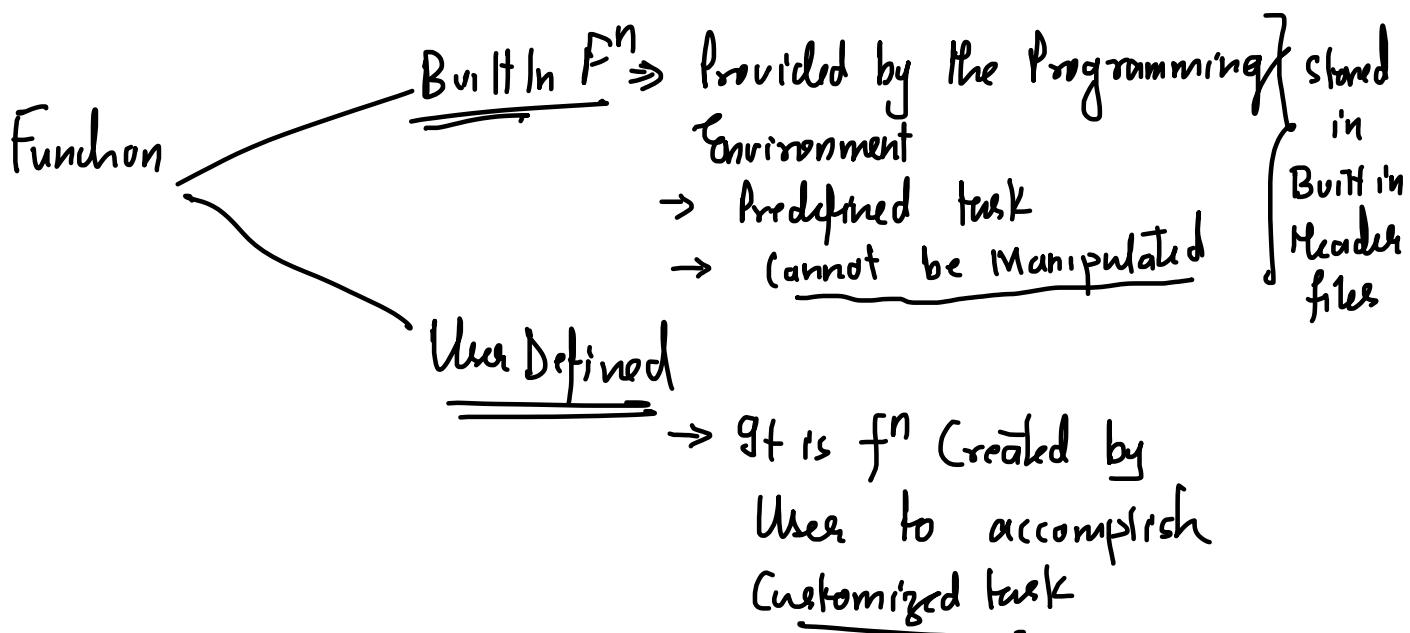
case 6:printf("exit condition\n");break;

default:printf("invalid choice\n");break;
}

}while(ch!=6);
return 0;
}
```

V-ImpFunction

- \* It is piece/block of code created to accomplish a defined task

Ex Consider

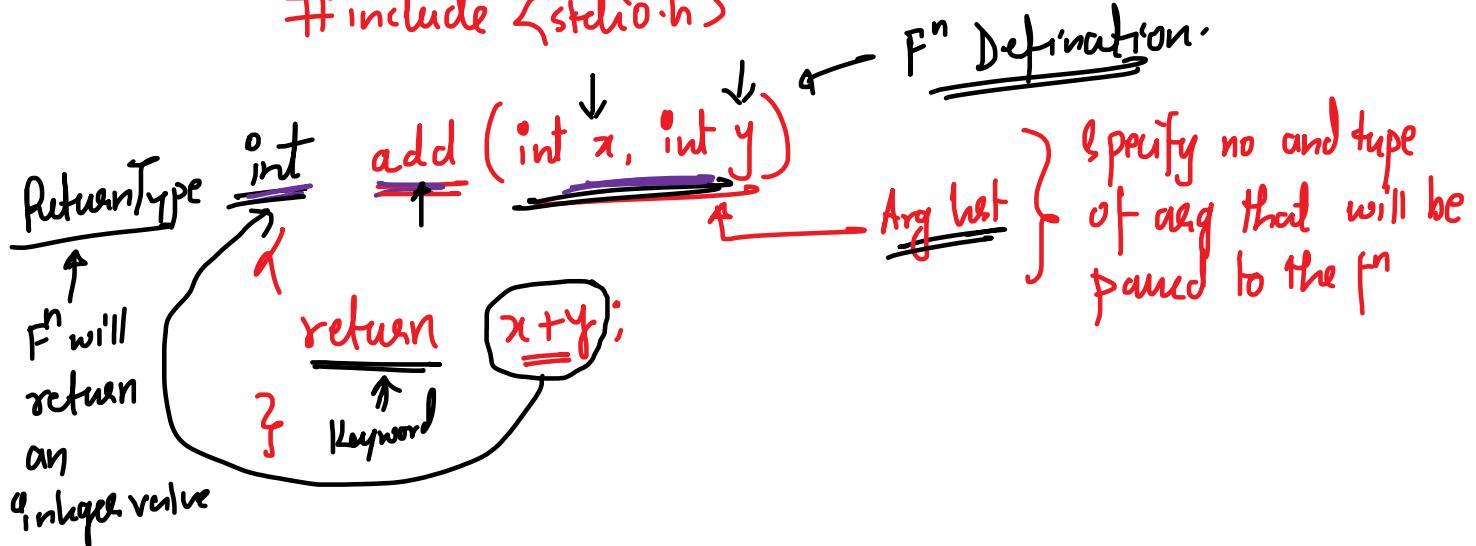
- \* C program to add 2 numbers without function

```
#include <stdio.h>
int main()
{
    int a, b, sum;
    printf("Enter two numbers\n");
    scanf("%d %d", &a, &b);
    sum = (a+b);
    printf("Sum = %d\n", sum);
```

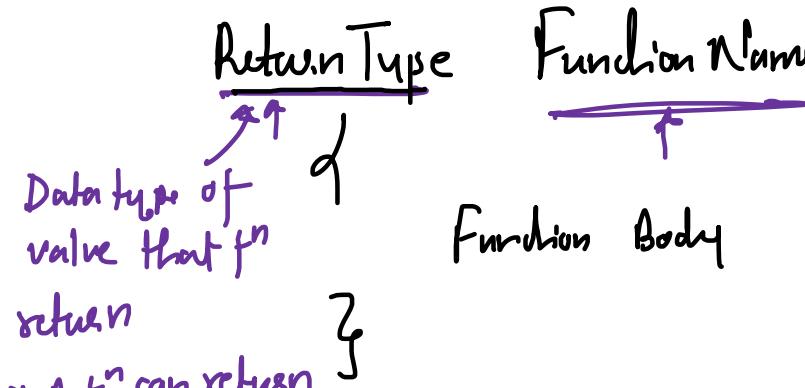
return 0;  
}

## \* Using function-

#include <stdio.h>



## F<sup>n</sup> Def<sup>n</sup> Syntax



\* A fn can return

only single argument.

\* There can be only one Return Type

formal arguments  
Function Name (type arg1, type arg2, ..., type argn)  
argument list  
( It specifies data type & No of arguments paired to the fn )

\* `#include <stdio.h>`

int add(int, int); →  $F^n$  Declaration / Prototype

int main() → called  $F^n$

{ int a, b, sum;

`printf("Enter two integer values\n");`

`scanf("%d %d", &a, &b),` values of a and b

sum = add(a, b); //  $F^n$  call  
                  ↑↑      actual argument

`printf("Sum = %d\n", sum);`

`return 0;`

}

int add(int x, int y) → called  $F^n$

{

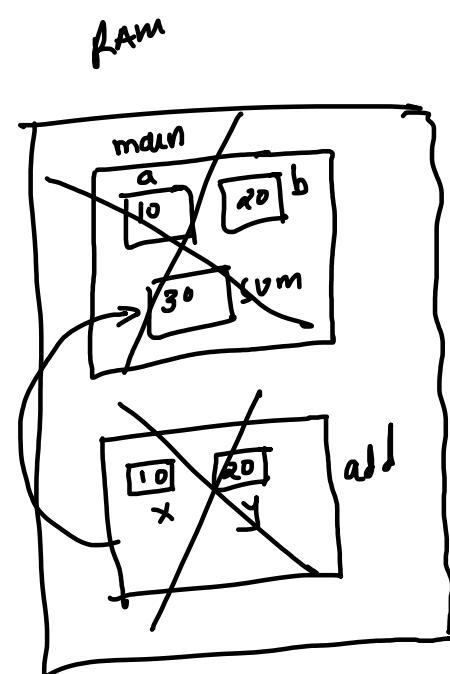
`return x+y;`

}

$F^n$

Definition

formal arg



In above Since the  $F^n$  Call appears before  $F^n$  Definition

the Compiler may generate Error

To avoid this

① Define  $F^n$  Before Call

② If  $f^n$  is defined after the call then

Declare the  $f^n$  before Call

Declare the  $f^n$  before call

\* Note >

① If  $f^n$  is defined after  $F^n$  call then  $f^n$  must be declared before function call.

② Formal Arguments  $\Rightarrow$  Arguments Specified in  $F^n$  definition are known as formal arguments.

③ Actual Arguments  $\Rightarrow$  Arguments Specified in  $F^n$  Call are known as actual arguments

④ A  $F^n$  can return only one argument (value/address) and the data type must be specified in ReturnType.

⑤ Function Call is replaced with value returned

⑥ Memory Allocation  $\Rightarrow$

① When a  $f^n$  is called, then memory <sup>(RAM)</sup> is allocated to the  $f^n$

② When a  $f^n$  Complete its Execution or it returns  $10$   $10$   $11$   $11$   $1n$  or required

then the memory assigned to the f<sup>n</sup> is regained

## Types of F<sup>n</sup>

① F<sup>n</sup> Taking arg & Returning value  $\Rightarrow$

```
#include <stdio.h>
int add(int, int);  $\rightarrow$  Fn Declaration
```

```
int main()
```

```
{ int a=10, b=20, sum;
```

```
sum = add(a, b);  $\rightarrow$  Fn call
```

```
printf("Result = %d\n", sum);
return 0,
```

}

```
int add(int x, int y)  $\rightarrow$  Fn Defn
```

```
{ return x+y;
```

}

---

② F<sup>n</sup> Taking Arg & Not Returning Value  $\Rightarrow$

```
#include <stdio.h>
```

```
void add(int, int);  $\rightarrow$  Fn Declaration
```

```
int main()
```

```
{ int a=10, b=20;
```

→ add(a, b);  $\rightarrow$  F<sup>n</sup> call  
return 0;

}

```
void add(int x, int y)  $\Rightarrow$  Fn Defn
```

{ printf("Result = %d\n", x+y);

}

---

③  $F^n$  Not Taking arg  
But Returns value  
to its caller

#include <stdio.h>  
 $\text{int add(void);} \Rightarrow F^n \text{ Declaration}$   
 $\text{int main() }$   
 $\{ \text{int sum;}$   
 $\text{sum = add();} \rightarrow F^n \text{ Call}$   
 $\text{printf("Result = %.d\n", sum);}$   
 $\text{return 0;}$   
 $\}$   
 $\text{int add(void) } \Rightarrow F^n \text{ Def }^n$   
 $\{ \text{int a=10, b=20, } \} \quad \text{scanf("%d %d", &a, &b),}$   
 $\text{return a+b; } \} \quad \text{return a+b;}$   
 $\}$

---

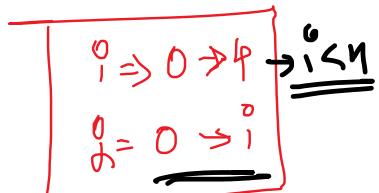
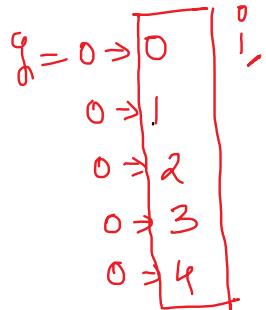
④  $F^n$  Not Taking arg  
& not returning value  
to its caller

#include <stdio.h>  
 $\text{void add();} \Rightarrow F^n \text{ Declaration}$   
 $\text{int main() }$   
 $\{ \text{add();} \rightarrow F^n \text{ Call}$   
 $\text{return 0;}$   
 $\}$   
 $\text{void add(void) } \rightarrow F^n \text{ Def }^n$   
 $\{ \text{int a=10, b=20, }$   
 $\text{printf("Result = %.d\n", a+b); } \}$

# Pascals Triangle

Q)

${}^0 C_0$	${}^1 C_0$	${}^1 C_1$	${}^2 C_0$	${}^2 C_1$	${}^2 C_2$	${}^3 C_0$	${}^3 C_1$	${}^3 C_2$	${}^3 C_3$	${}^4 C_0$	${}^4 C_1$	${}^4 C_2$	${}^4 C_3$	${}^4 C_4$
1														
	1													
		1												
			1											
				2										
					1									
						3								
							3							
								3						
									1					
										4				
											6			
												4		
													1	



Print  ${}^i C_j$

Note  ${}^i C_j \Rightarrow \underline{\underline{\text{comb}}}(\underline{\underline{i}}, \underline{\underline{j}})$

int c;

$$c = \frac{\underline{\underline{\text{fact}(i)}}}{\underline{\underline{(\text{fact}(j) \times \text{fact}(i-j))}}} ;$$

return c;

values

}

int fact(int n)

if  $n = 1$ ,  $f = 1$ ,

for ( $i = 1$ ;  $i \leq n$ ;  $i++$ )

$f = f \times i$ ;

return f;

}

#include<stdio.h>

```

int fact(int n)
{
    int i,f=1;
    for(i=1;i<=n;i++)
        f=f*i;

    return f;
}

int com(int i,int j)
{
    int c;
    c=(fact(i)/fact(j))/fact(i-j);
    return c;
}

```

auto, local to com

for Every call to com function  
The fact function is called  
3 times

```

int main()
{
    int i,j,n;
    printf("enter numbers of rows\n");
    scanf("%d",&n);
}

```

```

for(i=0;i<n;i++)
{
    for(j=0;j<=i;j++)
        printf("%3d",com(i,j));

    printf("\n");
}

return 0;
}

```

## Storage Classes (V-Imp)

\* Storage classes are assigned to the identifiers and it represents following about variables → variables

- ✓ ① Memory location
- ✓ ② Initial Value
- ✓ ③ Scope of Variable ⇒ Area of the program where the variable is accessible
- ✓ ④ Lifetime of Variable ⇒ when memory is assigned to variable and when memory is released

## Types of Storage Classes

- { ① auto (default)
- ② register
- ③ static
- ④ extern

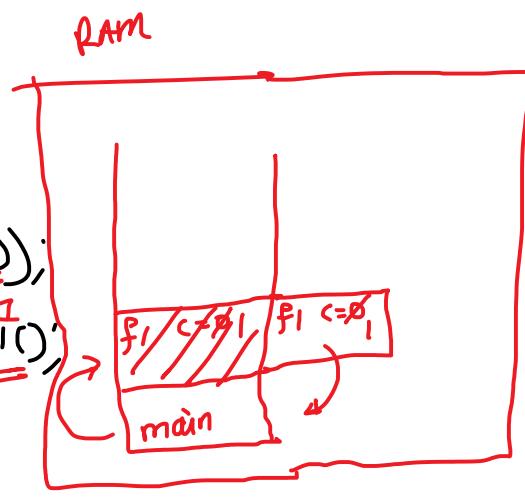
### ① auto Storage Classes

- \* It is default Storage Class
- \* Memory Allocation → RAM
- \* Initial Value → Garbage
- \* Scope → Scope is within the block where the variable is declared }  
(local to the block where declared)

\* lifetime → Memory is assigned when control enters  
the block and destroyed when control  
leaves the block

Ex #include<stdio.h>  
int f1()  
{  
 auto int c=0;  
 ↑  
 c++;  
 return c;  
}  
? ↗

```
int main()
{
    printf("c = %d", f1()); 1
    printf("c = %d\n", f1()); 1
    return 0;
} op 1 I
```



\* Function call's lifetime is managed using Stack Data Structure

\* Stack → It is data structure that operates in LIFO fashion.

\* For auto variable, the value is not retained between function call.

```
#include<stdio.h>
void f1()
{
    int c=0;
    printf("c = %d\n", c);
    c++;
}
int main()
{
    f1();
    f1();
    f1();
    return 0;
}
```

## Register storage class

\* Memory Allocation  $\Rightarrow$  In CPU register (so access is fast)  
 (Memory in Processor itself)

\* Initial Value = garbage

\* Scope  $\Rightarrow$  Within the block where declared

\* Lifetime  $\Rightarrow$  As long as the control is in function

```
#include<stdio.h>
void f1()
{
    register int c=0;  $\Rightarrow$ 
    printf("c= %d\n", c);
    c++;
}
int main()
{
```

f1();

f1();  $\rightarrow$

f1();  $\rightarrow$

return 0;

}

O/P

0

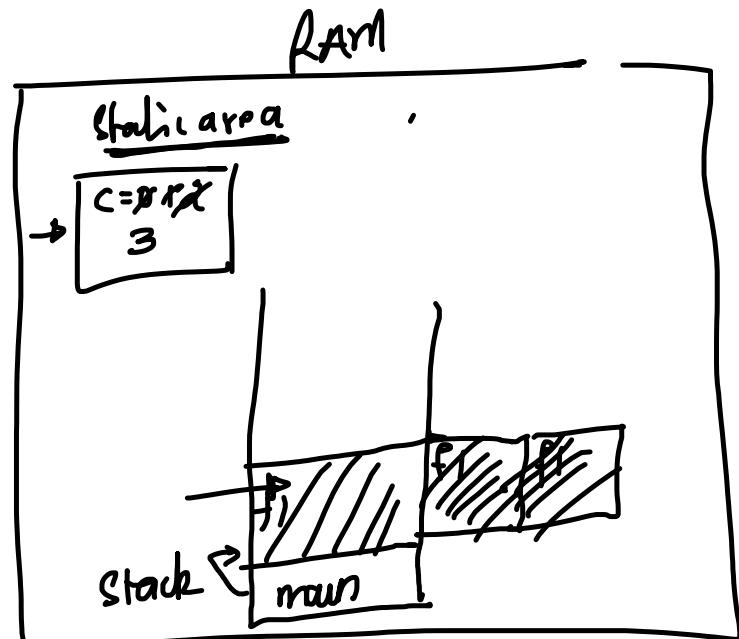
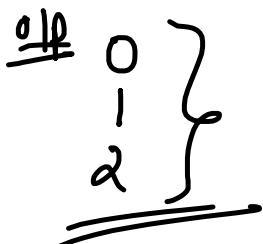
0

0

## Static Storage Class

- ① Memory Allocation = RAM
- ② Initial Value  $\Rightarrow$  0
- ③ Scope  $\Rightarrow$  within the block where declared
- ④ Lifetime  $\Rightarrow$  As long as the program is in execution
  - $\Rightarrow$  Value is retained bet<sup>n</sup> fn calls

```
#include<stdio.h>
void f1()
{
    static int c=0;
    printf("c= %d\n", c);
    c++;
}
int main()
{
    f1();
    f1();
    f1();
    return 0;
}
```



## extern storage class

→ It is used to tell compiler that variable defined as extern is declared with an external linkage elsewhere in program

✓ Storage2.c  
 not defined  
 ✓ extern int x; // declared as extern  
 ✓ void display()  
 {  
 printf("in storage 2 x=%d\n",x);  
 }

value of extern variable

✓ Storage3.c  
 #include<stdio.h> included a C program  
 ✓ #include"storage2.c"  
 ✓ int x=10; // necessary to re declare (defined)  
 global here  
 int main()  
 {  
 display();  
 printf("in storage 3 x=%d\n",x);  
 return 0;  
 }

{  
 In Storage 2 x=10  
 In Storage 3 x=10

- \* Variable declared as extern are not allocated any memory
- + (It is only declaration which specifies that variable is defined somewhere else)
- \* Default value is Zero
- \* Memory Allocation in RAM
- \* Scope → Across the program (since global)
- \* Lifetime → As long as the program is in execution.

## Recursion : (5-10 Mark)

- \* It is property of a function by virtue of which a function calls itself repeatedly within its own body
- \* A Recursive fn must satisfy following →
  - \* There must be a terminating condition.
  - \* The terminating condition must be met in finite number of steps
- \* Recursion is implemented using Stack.

### Disadvantage →

- ① Recursive fn takes more time as compared to iterative fn due to overhead of function call.
- ② Recursive fn needs more memory as compared to iterative fn

Advantage → There are some Algorithmic Approach that can be effectively implemented using Recursion

- Ex
- ① Divide & Conquer
  - ② Backtracking

## Ex To calculate factorial of a Number

### Iterative Approach:

→ Uses loop

$$5! = \underbrace{1 \times 2 \times 3 \times 4 \times 5}_{\text{product of all natural}}$$

No from 1 to 5

```
Logic: 
int i, f=1;
for (i=1; i<=5; i++)
    f=f*i;
```

\*  $f^n$  is called only once.

for any value of  $n$

### Recursive Approach:

$$5! \xrightarrow{120} 5 \times 4! \xrightarrow{24} 4 \times 3! \xrightarrow{6} 3 \times 2! \xrightarrow{2} 2 \times 1! \xrightarrow{1} 1 \times 0! \xrightarrow{1}$$

logic

$$\begin{aligned} f &= \underline{\underline{\text{fact}(5)}} \xrightarrow{120} 5 \times \underline{\underline{\text{fact}(4)}} \xrightarrow{24} 4 \times \underline{\underline{\text{fact}(3)}} \xrightarrow{6} 3 \times \underline{\underline{\text{fact}(2)}} \xrightarrow{2} 2 \times \underline{\underline{\text{fact}(1)}} \xrightarrow{1} 1 \times \underline{\underline{\text{fact}(0)}} \xrightarrow{1} \\ &\text{terminating} \end{aligned}$$

Ex

int fact(int n)

{  
if ( $n == 0$ )  
return 1;

else

return  $n * \underline{\underline{\text{fact}(n-1)}}$ ;

}

In fact "f" we are calling fact function again for diff value of  $n$ .

### Code

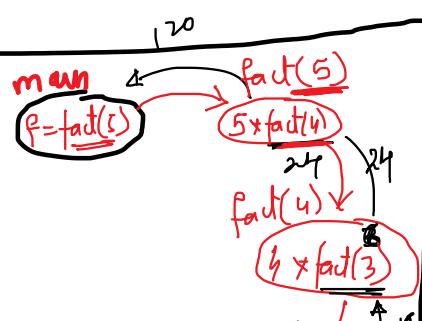
```
#include <stdio.h>
```

```
int fact(int n)
```

```
{ if ( $n == 0$ )
    return 1;
```

```
int main()
{
    int n, f;
    printf("Enter value of n\n");
    scanf("%d", &n);
    f = fact(n);
```

### RAM



```

    if(n==0)
        return 1;
    else
        return n * fact(n-1);
}

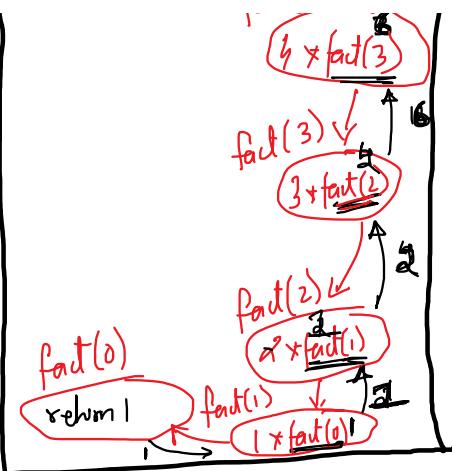
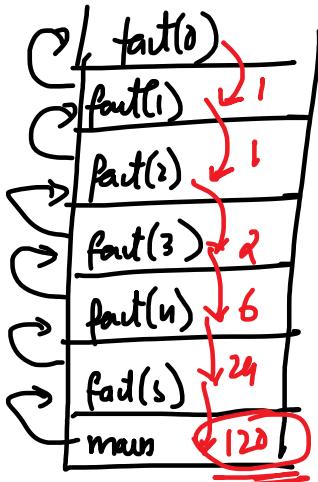
```

```

    f = fact(n);
    printf("Factorial = %d\n", f);
    return 0;
}

```

Stack Represent"



for n=5 fact() function  
is called 6 times  
n+1 time

~~Create a Recursive fn That returns factorial of a Number.~~

Use this fn to calculate  ${}^n P_r$  &  ${}^n C_r$

$$\frac{n!}{(n-r)!} = \frac{n!}{r! * (n-r)!}$$

```
#include<stdio.h>
```

```
int fact(int n)
{
    if(n==0)
        return 1;
    else
        return n*fact(n-1);
}
```

```
int main()
{
    int n,r;
    float p,c;

    printf("enter n and r\n");
    scanf("%d %d",&n,&r);

    p=(1.0*fact(n))/(fact(n-r));

    c=p/fact(r);

    printf("permutation value= %f\n",p);
    printf("combination value= %f\n",c);
    return 0;
}
```

Q) Write a recursive fn to find  $x+y$  and use it in main()

Ex  $\underline{3 \times 4} \Rightarrow 3$  is recursively added to itself 4 times.

```
#include <stdio.h>
int sum (int x, int y)
```

```
{ if (y == 0)
    return 0;
else
    return x + sum(x, y - 1);
```

{}

$$\begin{aligned}
 3 \times 4 &= \underline{\underline{3}} + \underline{\underline{3 \times 3}} \\
 &\downarrow \\
 &3 + \underline{\underline{3 \times 2}} \\
 &\downarrow \\
 &3 + \underline{\underline{3 \times 1}} \\
 &\downarrow \\
 &3 + \underline{\underline{3 \times 0}}
 \end{aligned}$$

```
int main()
```

```
{
    int a, b, r,
    printf("Enter two numbers\n"),
    scanf("%d %d", &a, &b);
```

$r = sum(a, b);$

```
printf("Result = %d\n", r);
return 0,
```

{}

Q) Write a recursive function to display sum of first n natural no

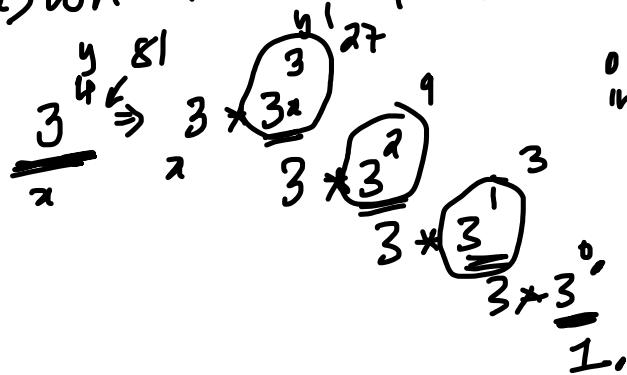
$$\begin{aligned} \underline{\text{Sum}}(5) &\Rightarrow 5 + \underline{\text{Sum}}(4) \\ &= 5 + 4 + \underline{\text{Sum}}(3) \\ &= 5 + 4 + 3 + \underline{\text{Sum}}(2) \\ &= 5 + 4 + 3 + 2 + \underline{\text{Sum}}(1) \\ &= 5 + 4 + 3 + 2 + 1 \end{aligned}$$

```
int sum(int n)
{
    if(n==1)
        return 1;
    else
        return n+sum(n-1);
}
```

```
#include<stdio.h>
int sum(int n)
{
    if(n==1)
        return 1;
    else
        return n+sum(n-1);
}
int main()
{
    int n;
    printf("enter n\n");
    scanf("%d",&n);

    printf("sum of first %d natural no= %d\n",n,sum(n));
    return 0;
}
```

Q) WAP Recursive f<sup>n</sup> to return  $x^y$  and use it in main().



int power(int x, int y)

{ if(y==0)  
return 1;

else  
return x \* power(x, y-1);

}

```
#include<stdio.h>
int power(int x,int y)
{
    if(y==0)
        return 1;
    else
        return x*power(x,y-1);
}
```

```
int main()
{
    int x,y;
    printf("enter the value of x and y\n");
    scanf("%d %d",&x,&y);

    printf("value of %d raised to %d = %d\n",x,y,power(x,y));
    return 0;
}
```

Q) Write a Recursive fn that returns reverse of a Number and use this fn to determine if a number is palindrome or not?

Logic Palindrome: If reverse of the number is original number itself, then it is known as palindrome.

1221  $\rightarrow$  Reverse 1221 so palindrome

reverse logic  $n = \underline{1234}$   $rev = 0$

$$\begin{aligned} & \text{Recursive} \\ & \left\{ \begin{aligned} r &= n \% 10 \Rightarrow 1234 \% 10 \Rightarrow 4 \\ rev &= rev * 10 + r \\ &\Rightarrow 0 * 10 + 4 \Rightarrow 4 \\ n &= n / 10 \Rightarrow 1234 / 10 \Rightarrow 123 \end{aligned} \right. \end{aligned}$$

$$\begin{aligned} r &= 123 \% 10 = 3 \\ rev &= 4 * 10 + r \\ &= 4 * 10 + 3 = 43 \\ n &= n / 10 = 123 / 10 \Rightarrow 12 \end{aligned}$$

$$\begin{aligned} r &= 12 \% 10 = 2 \\ rev &= 43 * 10 + 2 \\ &= 432 \\ n &= 12 / 10 = 1 \\ &= 3432 \\ n &= 1 / 10 = 0 \end{aligned}$$

Iterative logic  $\text{int main()}$

$\text{int } n, rev = 0, r, n1;$

`printf("Enter the number\n");  
scanf("%d", &n);`

In next iteration

$$n = n / 10$$

$$rev = rev * 10 + (n \% 10)$$

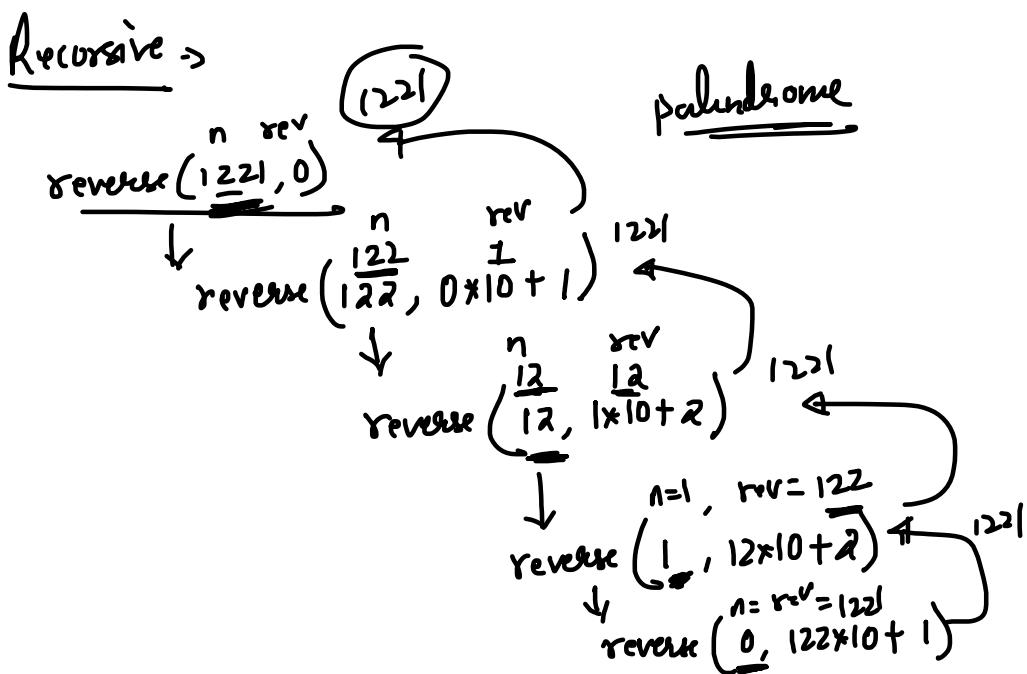
```
n1 = n;
while(n1 != 0)
{
    r = n1 % 10;
    rev = rev * 10 + r;
    n1 = n1 / 10;
}
```

Recursive logic

$n, rev = 0,$

if  $n == 0$   
return rev;  
else  
return

$\text{rev} = \underline{\text{rev}} \times 10 + (\underline{n} \cdot 10)$   
 $n1 = \underline{n1} / 10$ ,  
 if ( $\underline{\text{rev}} == n$ )  
 return  
 pri... (" $\underline{y}$ " is palindrome( $n$ ,  $n$ ));  
 else  
 printf("" $\underline{y}$ " is Not Palindrome( $n$ ,  $n$ ));  
 return 0;  
}



Q) Write a recursive fn that returns  $n^{\text{th}}$  term of fibonacci series

Use this fn to display first n terms of fibonacci series

term $\rightarrow$	1	2	3	4	5	6	7	8	9	10
Natural $\Rightarrow$	1	1	2	3	5	8	13	21	34	55
No	1	1	2	3	5	8	13	21	34	55
Serial	1	1	2	3	5	8	13	21	34	55

int febo( int n )

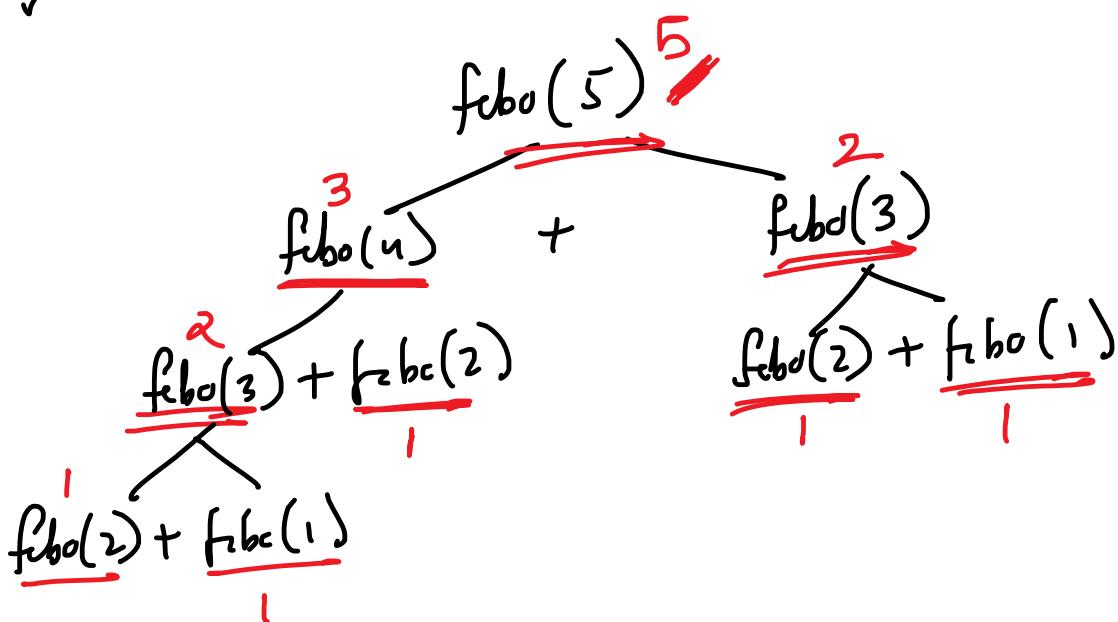
{     if ( n==1 || n==2 )

        return 1;

    else

        return febo(n-1) + febo(n-2);

}



#include<stdio.h>

```

int fibo(int n)
{
    if(n==1)
        return 0;
    else if(n==2)
        return 1;
    else
        return fibo(n-1)+fibo(n-2);
}

```

```

int main()
{
    int n,i;
    printf("enter the value of n\n");
    scanf("%d",&n);

```

```
printf("first %d term of fibo series= \n",n);
```

```

for(i=1;i<=n;i++)
    printf("%d\t",fibo(i));
return 0;
}

```

*First 6 term of fibo Series*

$$i=1 \quad \underline{fibo(1)} \rightarrow 0$$

$$i=2 \quad fibo(2) \rightarrow 1$$

$$i=3 \quad fibo(3) \rightarrow 2$$

$$i=4 \quad fibo(4) = \rightarrow 3$$

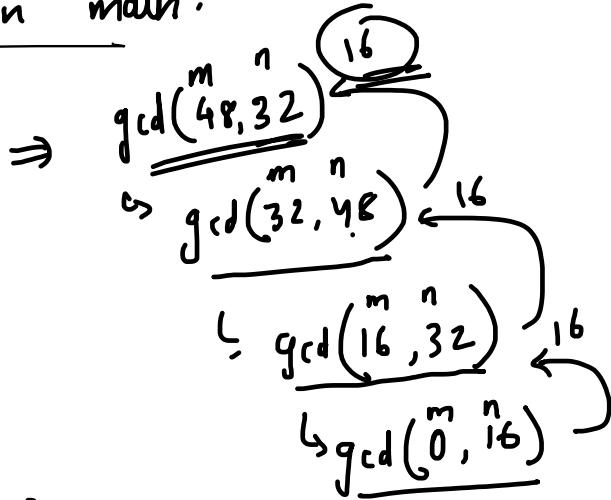
$$i=5 \quad fibo(5) = \rightarrow 5$$

$$i=6 \quad fibo(6) = \rightarrow 8$$

Q) Write a recursive fn that returns gcd of two numbers

Use this fn in main.

```
#include<stdio.h>
int gcd(int m,int n)
{
    if(m==0)
        return n;
    else if(n>m)
        gcd(n%m,m);
    else
        gcd(n,m); — Swap m & n.
}
int main()
{
    int m,n,l,g;
    printf("enter the value of m and n\n");
    scanf("%d %d",&m,&n);
    g=gcd(m,n);
    l=(m*n)/g;
    printf("gcd of %d and %d is %d\n",m,n,gcd(m,n));
    printf("lcm of %d and %d is %d\n",m,n,l);
    return 0;
}
```



$$48 = 1, 2, 4, 3, 6, 8, 12, 16,$$

24, 48

$$32 = 1, 2, 4, 8, 16, 32$$

Common →

$$1, 2, 4, 8 \quad \text{circled } 16$$

$$\text{gcd value} = 16$$

Logic Let m & n be two numbers and let g & l be the gcd & lcm then

$$m \times n = g \times l$$

$$l = \frac{m \times n}{g}$$

Q) Write a recursive fn that returns sum of digits of a Number like it is meant to -

Logic

Iterative logic  $\rightarrow$

```

int n, sum=0;
int y;
while (n > 0)
{
    y = n % 10;
    sum = sum + y;
    n = n / 10;
}

```

$$\text{Ex} \quad n = \underline{\underline{5486}}$$

$$\begin{aligned}
 y &= 5486 \% 10 = \underline{\underline{6}} \\
 \text{sum} &= 0 + 6 = \underline{\underline{6}} \\
 n &= 5486 / 10 = \underline{\underline{548}}
 \end{aligned}$$

$$\left. \begin{aligned}
 y &= 548 \% 10 = \underline{\underline{8}} & y &= 54 \% 10 = \underline{\underline{4}} \\
 \text{sum} &= 6 + 8 = \underline{\underline{14}} & \text{sum} &= 14 + 4 = \underline{\underline{18}} \\
 n &= 548 / 10 = \underline{\underline{54}} & n &= 54 / 10 = \underline{\underline{5}} \\
 \end{aligned} \right\} \quad \begin{aligned}
 y &= 5 \% 10 = \underline{\underline{5}} \\
 \text{sum} &= 18 + 5 = \underline{\underline{23}} \\
 n &= 5 / 10 = \underline{\underline{5}} / 10 = \underline{\underline{0}}
 \end{aligned}$$

Recursive logic  $\rightarrow$

sum = sum + (n % 10);      while  $n \neq 0$   
 $n = n / 10;$       if  $n=0$  return sum

```

#include<stdio.h>
int sum(int n, int s)
{
    if(n==0)
        return s;
    else
        sum(n/10,s+n%10);

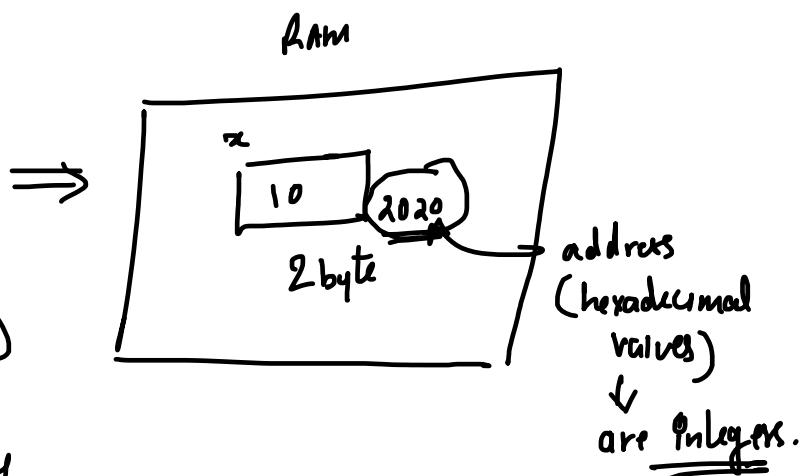
}
int main()
{
    int n,s;
    printf("enter the number\n");
    scanf("%d",&n);
    s=sum(n,0);
}

```

```
printf("sum of digit of %d is %d\n",n,s);
return 0;
}
```

Pointers  $\Rightarrow$

Note  $\text{int } x;$   
 $\uparrow$   
 Identifier (Name of Memory)



In above for a byte of Memory

Name of the Memory  $\Rightarrow \underline{x}$

Address of the Memory  $\Rightarrow \underline{2020}$

To Access any Memory location

Need  
 Using its name ✓  
 OR  
 Using its address ✓

Uptill Now We used Name of the Memory (Using Identifier)

To assign value  $\rightarrow \text{scanf("y.d", } \underline{\underline{x}});$   
 $\uparrow$  address of  $x$  (2020) Here

To print value  $\rightarrow \text{printf("y.d", } \underline{\underline{x}});$   
 $\uparrow$  value at memory  $\underline{x}$   
 $\downarrow$  value at memory  $\underline{2020}$

Pointers  $\rightarrow$  Pointers are used to access Memory using its address.

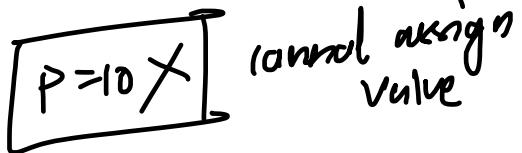
\* Pointer variable will never store value, will always store address  
Note  $\text{int } p;$   $\rightarrow$  Memory identified by  $p$  will store integer value

Note  $\text{int } p;$   $\rightarrow$  Memory identified by  $p$  will store integer value

## Pointer Declaration $\Rightarrow$

Type  $* \text{pointername};$

Ex  $\boxed{\text{int } * p;}$   $\Rightarrow$  Integer pointer



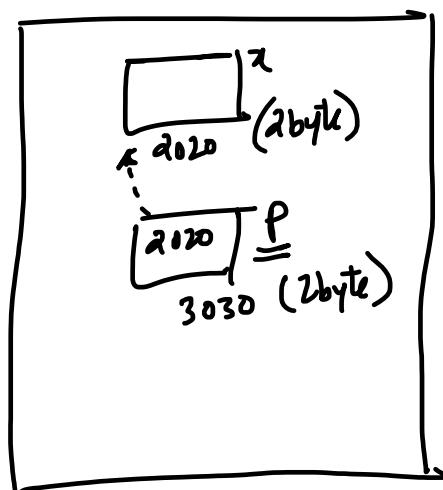
"P is a pointer variable, it will only store address, here the type of pointer is int, the pointer p will store address of int memory only"

RAM

Note  $\Rightarrow$   $\text{int } x;$   $\Rightarrow$  Integer variable  
 $\rightarrow \boxed{\text{int } * p;}$   $\Rightarrow$  Integer pointer  
 $\boxed{\text{float } * q;}$   $\Rightarrow$  Float pointer  
 $\downarrow$  \* will store address of float memory

$\boxed{q = \&x;}$  X  
we cannot assign address of int x to float pointer q

$p = \&x;$  ✓



Pointers of any type will store address only

Address of any type memory is hexadecimal value.

(integer)

- \* So every pointer needs 2 bytes only to store address
- \* So size of every pointer is 2 bytes.

(List of  
address)      int \*p;  
                  float \*p;  
                  char \*p;  
                  long \*p;

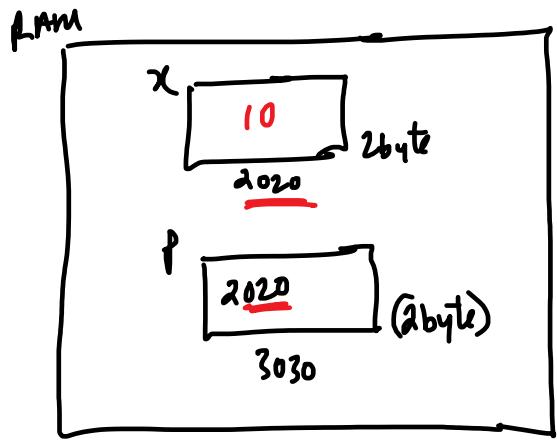
}      size = 2 bytes

Consider

```
int x;
int *p;
```

- // To make a pointer point / refer to a Memory
- // assign address of the memory to the pointer

$$p = \underline{\&}x;$$



// Now since pointer is referring to the memory

// we can access the memory using pointer

without pointer

To assign value.

```
scanf("%d", &x);
      2020
```

with Pointer

scanf("%d", p); → address of memory pointed by p  
2020

To print value.

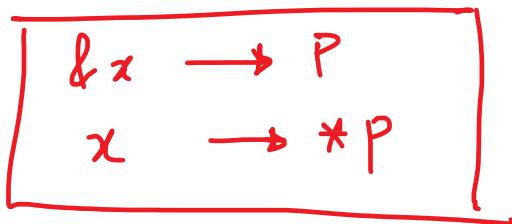
```
printf("%d", x);
      ↓
      value at address 2020
      => 10
```

printf("%d", \*p); → value at memory pointed by p  
\*p,  
\*(2020),  
value at address 2020

MatLab →

Ans 1. `int x;`  
And 2. `int *p = &x;`

Then



If "P" is a pointer (of any type)

Then  $P \Rightarrow$  will give address of memory to which pointer  
is pointing (P jisko point karta hua uska address)

$*P \Rightarrow$  value/content at memory to which pointer  
is pointing  
(P jisko point karta hua uska value).

Note  $\&$   $\Rightarrow$  Reference operator (Assign address)

$*$   $\Rightarrow$  Dereference operator (Assign value at address)

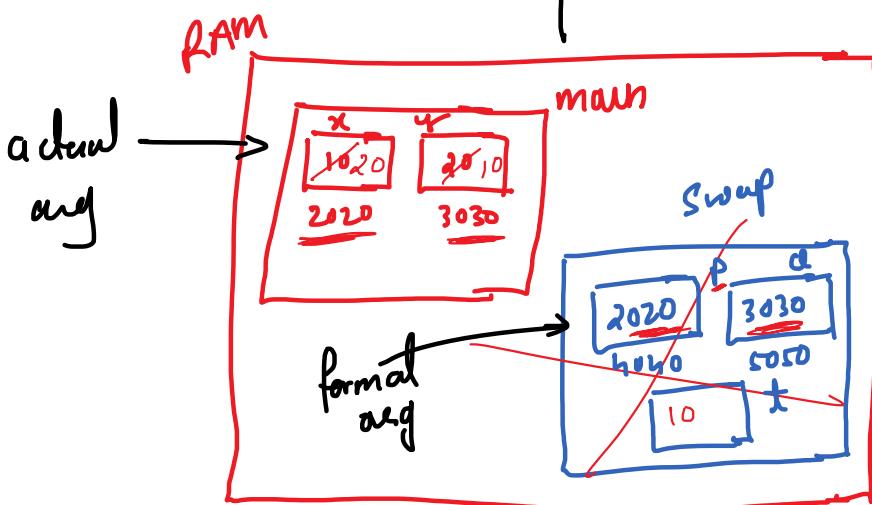


## Call By Reference $\Rightarrow$ Value

Consider

```
#include<stdio.h>
void swap(int *p, int *q)
{
    int t;
    t = *p;  $\Rightarrow$  t = *(2020)
    *p = *q;  $\Rightarrow$  * (2020) = *(3030)
    *q = t;  $\Rightarrow$  *(3030) = 10
}
```

```
int main()
{
    int x=10, y=20;
    printf("Before Calling Swap function\n");
    printf("x=%d \n y=%d \n", x, y);
    → swap(&x, &y);  $\downarrow$  address of x & y
    printf("After calling Swap function\n");
    printf("x=%d \n y=%d \n", x, y);
    return 0;
}
```

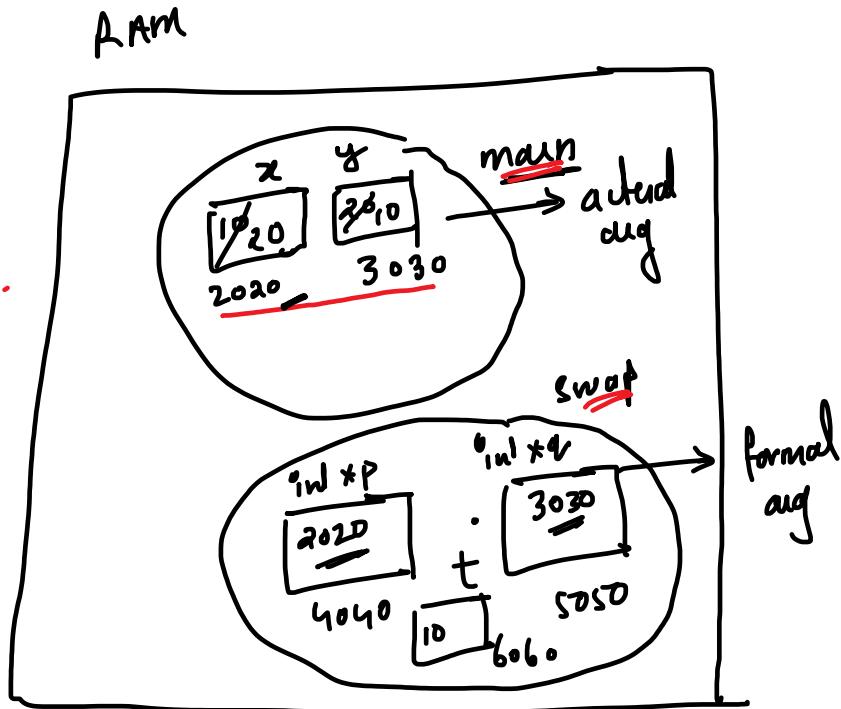
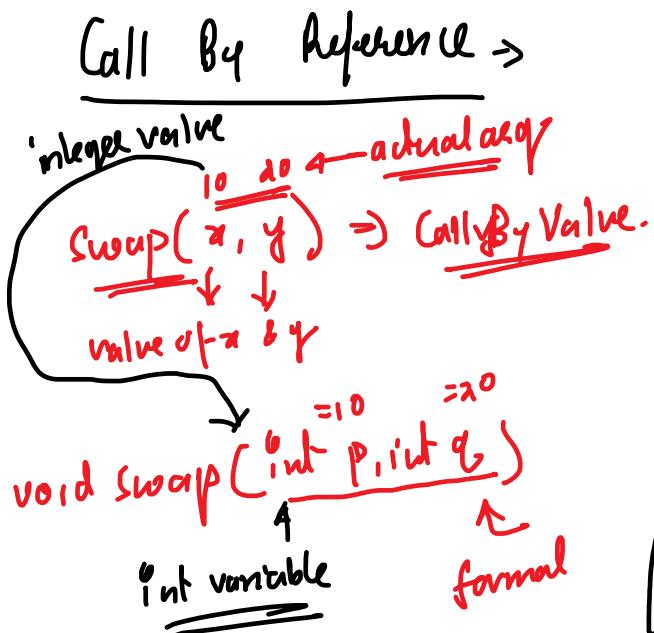


Q1P  
Before Calling Swap function  
 $x=10$   
 $y=20$   
After Calling Swap function  
 $x=20$   
 $y=10$

[In call By Reference, the actual arg passed in function call are address and the formal arg specified in function definition are pointers that refer to these address.]

\* The actual arg & formal arg refers to same Memory location.

\* Hence changes made by formal arg are reflected in  
actual arg



Call By Reference

swap(  $\underline{2020}$ ,  $\underline{3030}$  ) ; } } Address of  $x \& y$

(address of int  $\leftarrow$  actual arg  
value)

So void Swap(int \*p, int \*q)

q    $\Leftarrow$  The Swap f<sup>n</sup> now has address  
of memory inside main()

\* The swap f<sup>n</sup> does not have name of the memory  
locations 2020 & 3030 of main f<sup>n</sup>

\* Now Swap f<sup>n</sup> has add of Memory location in main f<sup>n</sup>  
 & now these address can be accessed using pointer

void swap(<sup>int</sup>  $\&P$ , <sup>int</sup>  $\&q$ )  
 {  
     int t;  
     t = \*P;  
     \*P = \*q;  
     \*q = t;  
 }

$t = *P = *(\underline{\underline{2020}}) \Rightarrow 10$   
 $*(\underline{\underline{2020}}) = *(\underline{\underline{3030}}),$   
 $*(\underline{\underline{3030}}) = 10$

Array  $\rightarrow$  It is Contiguous collection of elements of Same type

↓

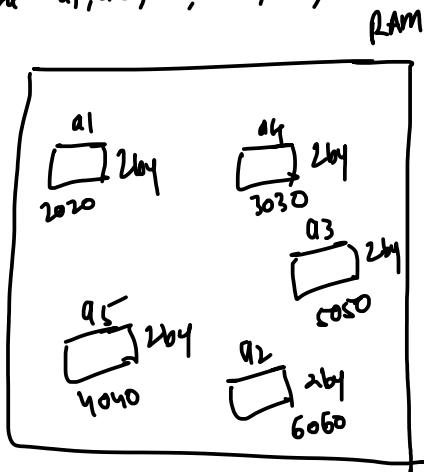
Contiguous  
 Complete

Contiguous  $\rightarrow$  Continuous memory allocation

Complete  $\rightarrow$  Not Shareable, totally assigned to single array

Syntax  $\rightarrow$  type arrayname [size];

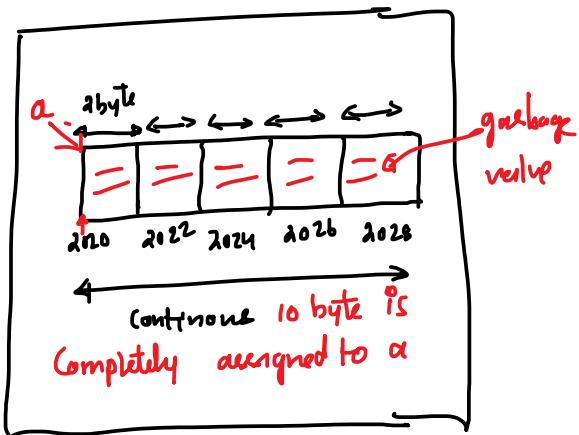
`int a1, a2, a3, a4, a5;`



Total Allocation =  $2 \times 5 = 10$  bytes  
 $\downarrow$  random (Not Contiguous)

Create an array of 5 integers

int a[5]; RAM



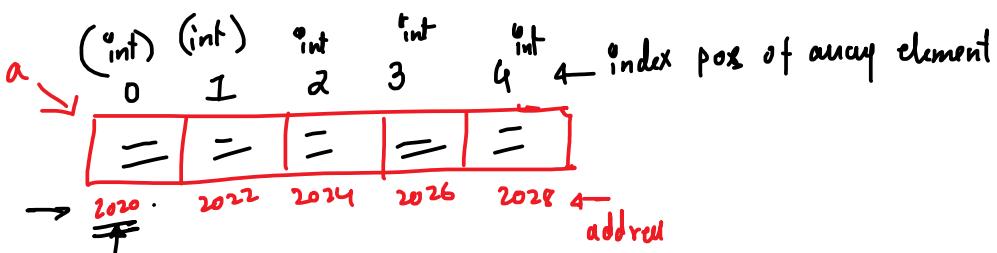
New  $\rightarrow$

`int a[5];`

\* address of first element of array is known as base address of array

\* array name indicates base address.

$$a \equiv 2020$$



\* The index "pos" represents distance of the element from base  $\Rightarrow$

0  $\Rightarrow$  at distance of 0 until to 0 byte from base

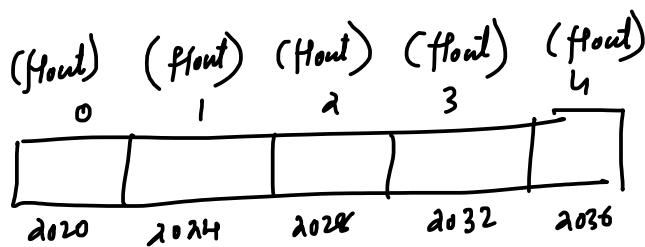
1  $\Rightarrow$  n " " 1 int is 2 byte " "

2  $\Rightarrow$  " " 2 " " 4 byte " "

3  $\Rightarrow$  " " 3 " " 6 byte " "

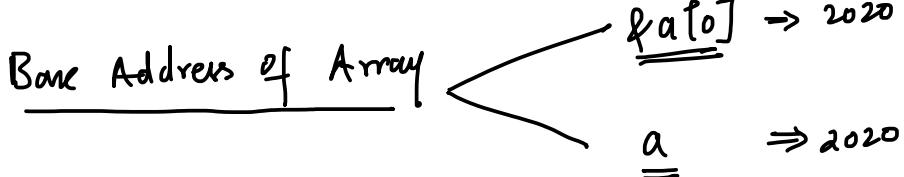
Note

float a[5];



\* The array is default initialized with garbage value

Note Index of base element of array  $\Rightarrow 0$

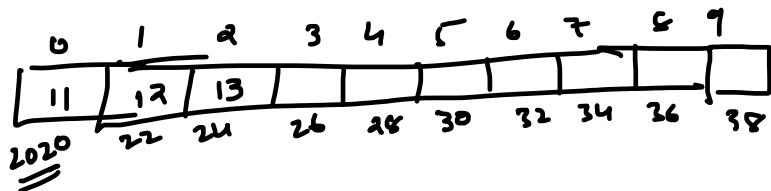


Q) Write a program to fill 10 integers in an array and display them.

```
#include <stdio.h>
```

```
int main()
```

```
{ int a[10];
```



```
printf("Enter the elements in array\n");
```

```
for(i=0; i<10; i++)
    scanf("%d", &a[i]);
```

$$\begin{array}{ll} i=0 & \& a[0] = 2020 \\ i=1 & \& a[1] = 1122 \\ i=2 & \& a[2] = 2024 \\ & \vdots \end{array}$$

```
printf("Elements in array are\n");
```

```
for(i=0; i<10; i++)
    printf("%d\n", a[i]);
```

$\Rightarrow$  int

$$\begin{array}{l} i=0 \quad a[0] \Rightarrow *(\underline{\underline{a[0]}}) = \\ \quad \quad \quad *(\underline{\underline{2020}}) = 11 \end{array}$$

$$\begin{array}{l} i=1 \quad a[1] = *(\underline{\underline{a[1]}}) \Rightarrow 12 \end{array}$$

```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
    int a[10], i;
```

```
    printf(" enter elements in array\n");
```

```
    for(i=0;i<10;i++)
```

```
        scanf("%d",&a[i]);
```

```
printf(" elements in array are\n");
for(i=0;i<10;i++)
printf("%d\t",a[i]);

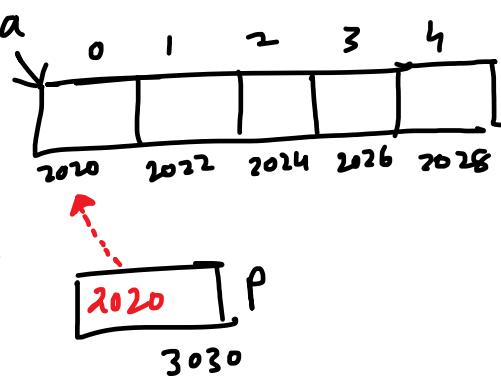
return 0;
}
```

## Array Using Pointer :-

\* We can access an array using pointer.

Step1 → Create an array  $\Rightarrow$  `int a[5];`

Step2 → Create a pointer of same type as that of array  $\Rightarrow$  `int *p;`



Step3 → Assign base address of array to the pointer  $\Rightarrow$

$P = \underline{\underline{&a[0]}};$  } assign address of first  
OR       $P = \underline{\underline{a}};$  } element of array

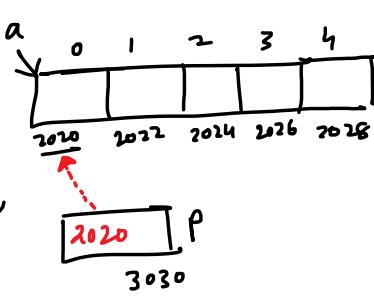
\* When pointer is initialized with array base address we say the pointer is pointing to the ~~addr~~ array

## Array Using Pointer

\* We can access an array using pointer.

Step 1 → Create an array  $\Rightarrow \text{int } a[5];$

Step 2 → Create a pointer of same type as that of array  $\Rightarrow \text{int } *p;$



Step 3 → Assign base address of array to the pointer  $\Rightarrow$

$p = \&a[0];$  } assign address of first  
or  $p = a;$  } element of array

\* When pointer is initialized with array base address we say the pointer is pointing to the ~~array~~ array

\* We made a pointer point to an array so that now we can access the array using pointer

## To Read Element in Array

### Without pointer

`for(i=0; i<5; i++)`

`scanf("%d", &a[i]);`

$$i=0 \quad \&a[0] = 2020$$

$$i=1 \quad \&a[1] = 2022$$

$$i=2 \quad \&a[2] = 2024$$

$$i=3 \quad \&a[3] = 2026$$

$$i=4 \quad \&a[4] = 2028$$

### Using Pointer

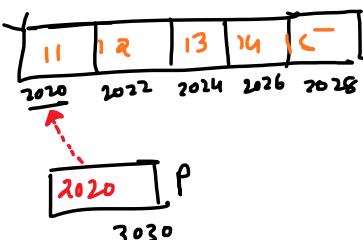
`for(i=0; i<5; i++)`

`scanf("%d", (p+i));`

$$i=0 \quad p+i \Rightarrow 2020 + 0(\text{int}) = \\ \Rightarrow 2020 + 0 \Rightarrow 2020$$

$$i=1 \quad p+i \Rightarrow 2020 + 1(\text{int}) \\ \Rightarrow 2020 + 1 = 2022$$

$$i=2 \quad p+i \Rightarrow 2020 + 2(\text{int}) \\ \Rightarrow 2020 + 2 = 2024$$



## To Display the Content of the Array

### Without pointer

```
for(i=0; i<5; i++)
printf("%d\t", a[i]);
i=0 a[0] = *(&a[0]) = *(2020) = 11
i=1 a[1] = *(&a[1]) = *(2022) = 12
i=2 a[2] = *(&a[2]) = *(2024) = 13
i=3 a[3] = *(&a[3]) = *(2026) = 14
i=4 a[4] = *(&a[4]) = *(2028) = 15
```

### With pointer

```
for (i=0; i<5; i++)
printf("%d\t", *(p+i));
i=0 => *(p+0) = *(2020) = 11
i=1 => *(p+1) = *(2022) = 12
i=2 => *(p+2) = *(2024) = 13
i=3 => *(p+3) = *(2026) = 14
i=4 => *(p+4) = *(2028) = 15
```

```
#include<stdio.h>
int main()
{
    int a[10], i;
    int *p;

    p=a;//assigned base address of array to the pointer

    printf(" enter elements in array\n");

    for(i=0;i<10;i++)
        scanf("%d", (p+i));

    printf(" elements in array are\n");
    for(i=0;i<10;i++)
        printf("%d\t", *(p+i));

    return 0;
}
```

Conclusion  $\Rightarrow$

if  $\text{int } a[5];$   
& if  $\text{int } *p = a;$

$$\begin{aligned}\text{Then } & \&a[i] \Rightarrow (p+i) \\ & a[i] \Rightarrow *(p+i) \\ & \underline{\underline{\qquad\qquad\qquad}}\end{aligned}$$

"In above program, the  $f^n$  main() has name of the array so it does not make any sense to create a pointer & access the array using pointer "

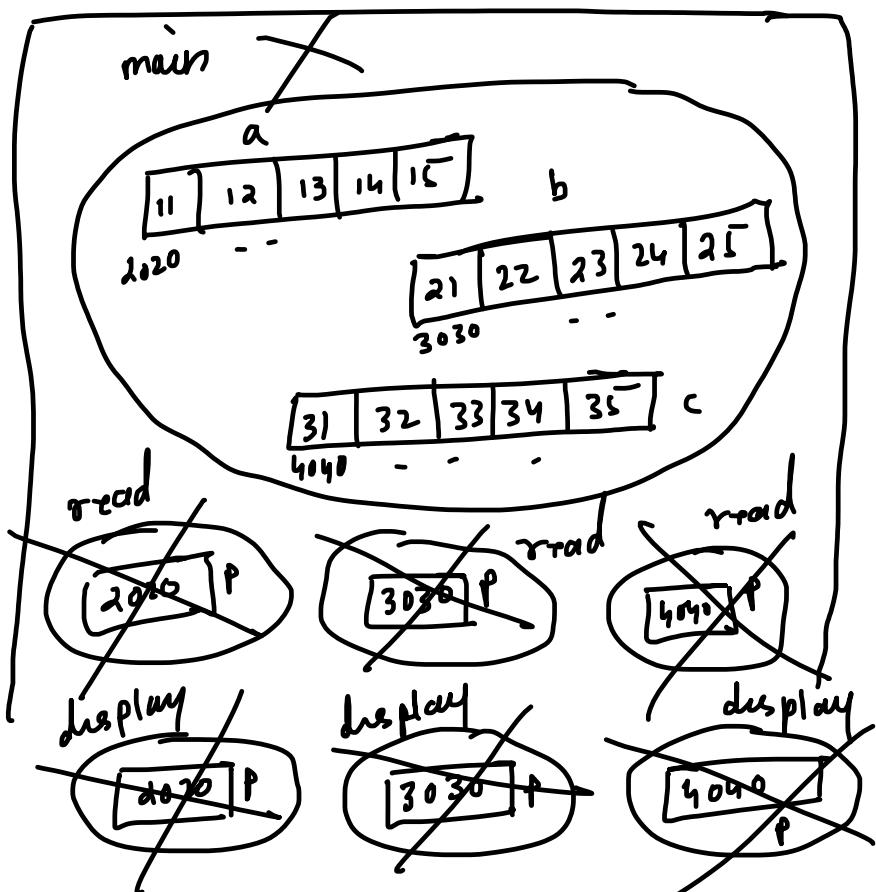
→ We ideally use pointer when we do not have access to the name but only address

# \* Most Important Concept in C Programming

## \* Passing array to function

Dhyan do !!

```
#include<stdio.h>
= 2020 3030 4040
void read(int *p)
{
    int i;
    for(i=0;i<5;i++)
        scanf("%d",*(p+i));
}
2020
void display(int *p)
{ int i;
    for(i=0;i<5;i++)
        printf("%d\t",*(p+i));
    printf("\n");
}
int main()
{
    int a[5],b[5],c[5]; ✓
```



~~o/p~~ elements in array a are  
11 12 13 14 15

printf(" enter elements in array a\n");
read(a); // read(&a[0])  $\Rightarrow$  read(2020);

✓ printf(" enter elements in array b\n");
read(b); // read(&b[0])  $\Rightarrow$  read(3030);

✓ printf(" enter elements in array c\n");
read(c); // read(&c[0])  $\Rightarrow$  read(4040);

elements in array b are  
21 22 23 24 25

elements in array c are  
31 32 33 34 35

read(c); // read(&c[0])  $\Rightarrow \underline{\text{read(4040);}}$

printf(" elements in array a are\n");  
display(a); // display(&a[0])  $\Rightarrow \underline{\text{display(2020)}}$

printf(" elements in array b are\n");  
display(b); // display(&b[0])  $\Rightarrow \underline{\text{display(3030)}}$

printf(" elements in array c are\n");  
display(c); // display(&c[0])

}

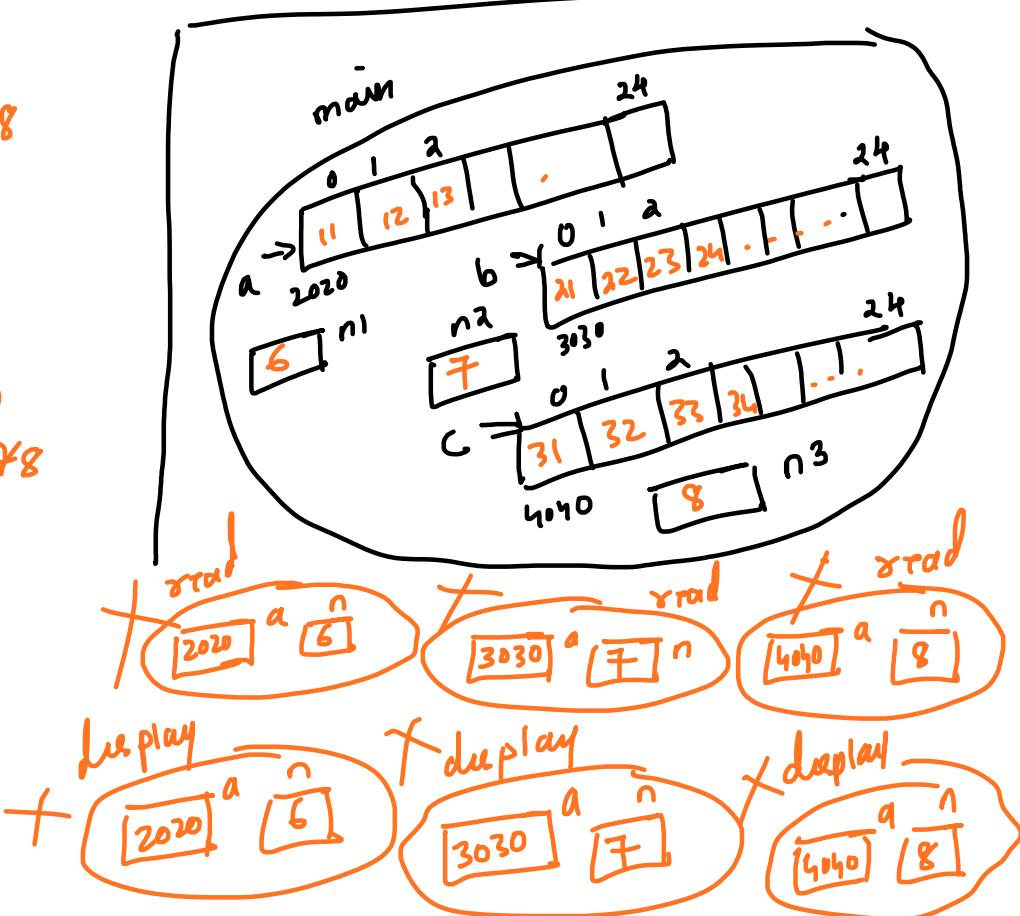
return 0;

When we want to perform operation on Many no of arrays  
So we pass array to the function

```
#include<stdio.h>
void read(int a[],int n)
{
    int i;
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
}
void display(int a[],int n)
{
    int i;
    for(i=0;i<n;i++)
        printf("%d\t",a[i]);
    printf("\n");
}
int main()
{
    int a[25],b[25],c[25];
    int n1,n2,n3;

    printf(" enter number of elements in array a\n");
    scanf("%d",&n1);

    printf(" enter elements in array x\n");
}
```



```
scanf(" enter number of elements in array a\n");
scanf("%d",&n1);
```

```
printf(" enter elements in array x\n");
```

```
read(a,n1); // read(&a[0],n1)  $\Rightarrow$  read(2020, 6)
```

```
printf(" enter number of elements in array b\n");
scanf("%d",&n2);  $\Rightarrow$  7
```

```
printf(" enter elements in array b\n");
```

```
read(b,n2); // read(&b[0],n2)  $\Rightarrow$  read(3030, 7)
```

```
printf(" enter number of elements in array c\n");
scanf("%d",&n3);  $\Rightarrow$  8
```

```
printf(" enter number of elements in array c\n");
scanf("%d",&n3);  $\Rightarrow 8$ 

printf(" enter elements in array c\n");

read(c,n3); // read(&c[0],n3)  $\Rightarrow$  read(4040,8)

printf(" elements in array a are\n");
display(a,n1); // display(&a[0],n1)  $\Rightarrow$  display(2020,6)

printf(" elements in array b are\n");
display(b,n2); // display(&b[0],n2)  $\Rightarrow$  display(3030,7)

printf(" elements in array c are\n");
display(c,n3); // display(&c[0],n3)  $\Rightarrow$  display(4040,8)

return 0;
}
```

X

Q) C program to find largest element in array.

Logic

Initially  $\rightarrow$

$$\begin{array}{|c|c|c|c|c|} \hline & 0 & 1 & 2 & 3 & 4 \\ \hline \text{max} & = a[0] & = \underline{\underline{10}} & & & \\ \hline \text{pos} & = 0 & & & & \\ \hline \end{array}$$

$n=5$	↓	2	3	4	
0	1	15	18	20	16

$$\begin{array}{l} i=1 \quad \text{is } a[1] > \text{max} \quad \text{Yes} \\ \text{max} = a[1] = \underline{\underline{15}} \\ \text{pos} = 1 \end{array}$$

$$\begin{array}{l} i=2 \quad \text{is } a[2] > \text{max} \quad \text{Yes} \\ \text{max} = a[2] = \underline{\underline{18}} \end{array}$$

pos = 2

$$\begin{array}{l} i=3 \quad \text{is } a[3] > \text{max} \quad \text{Yes} \end{array}$$

$$\begin{array}{|c|} \hline \text{max} = a[3] = \underline{\underline{20}} \\ \text{pos} = 3 \\ \hline \end{array}$$

$$\begin{array}{l} i=4 \quad \text{is } a[4] > \text{max} \quad \text{No} \\ \text{Nothing changes} \end{array}$$

```
#include<stdio.h>
```

```
void main()
{
```

```
int a[20];
```

```
int n,i,max,p;

printf("enter no of elements\n");
scanf("%d",&n);

printf("enter the elements in array\n");

for(i=0;i<n;i++)
scanf("%d",&a[i]);

printf(" array elements are\n");
for(i=0;i<n;i++)
printf("%d\t",a[i]);

printf("\n");

//logic
max=a[0];// let first element be largest
p=0;

for(i=1;i<n;i++)
{
    if(a[i]>max)
    {
        max=a[i];
        p=i;
    }
}

printf("largest value is %d at %d index position\n",max,p);

return 0;
}
```

## Q) Smallest Element in logic

$\min = a[0]; \checkmark$

$p = 0; \checkmark$  gives pos<sup>n</sup> of smallest element

for ( $i=1; i < n; i++$ )

if ( $a[i] < \min$ )  $\rightarrow$  if any  $i^{th}$  element ~~is~~ (at min

$\min = a[i]; \rightarrow$  new  $\min =$  thus  $i^{th}$  element

$p = i;$

}

}

printf("Smallest Element is %.d at position %d\n",  $\min, p$ );

VIMP → To Sort an array in Ascending Order

( Bubble Sort Technique )

Logic       $i = 5$

15	14	13	12	11
0	1	2	3	4

Sorted array in  
Ascending Order

11	12	13	14	15
0	1	2	3	4

Desired o/p

i.e. Here Condition to be  
Satisfied

if (any element > element to its right)  
& then swap both.  
?

i.e. if ( $a[j] > a[j+1]$ )  
 $t = a[j];$       } Swap  
 $a[j] = a[j+1];$   
 $a[j+1] = t;$       }

Here  
No element is greater than  
the element on its right  
(immediate)

13	12	11		
14	15	13	12	15
0	1	2	3	4

1st Iteration.

$j = 0$

$a[0] > a[1]$  Yes

Swap

$j = 1 \quad a[1] > a[2]$  Yes Swap

$j = 2 \quad a[2] > a[3]$  Yes Swap

$j = 3 \quad a[3] > a[4]$  Yes Swap

End of first Iteration.

After 1st Iteration  
→ largest element at bottom

13	14	13	12	15
0	1	2	3	4

2nd Iteration

$j = 0 \quad a[0] > a[1]$  Yes Swap

12	13	13	14	15
0	1	2	3	4

3rd Iteration →

$j = 0 \quad a[0] > a[1]$  Yes Swap

$j = 1 \quad a[1] > a[2]$  Yes Swap

11	12	13	14	15
0	1	2	3	4

4th Iteration

$j = 0 \quad a[0] > a[1]$  Yes Swap

$i=0$   $a[0] > a[1]$  Yes Swap

$i=1$   $a[1] > a[2]$  Yes Swap

$i=2$   $a[2] > a[3]$  Yes Swap

$i=3$   $a[3] > a[4]$  false

```

j=0 a[0] > a[1] Yes swap
j=1 a[1] > a[2] Yes Swap
j=2 a[2] > a[3] NO
j=3 a[3] > a[4] NO

```

$i = 0 \quad a[i] > a[j] \text{ yet Swap}$

Sorted Ascending Order

for ( $i = 0$ ;  $i < n - 1$ ;  $i++$ ) i=5  
0 44 n time

for( $j=0$ ;  $j < n-1$ ;  $j++$ )  $\Rightarrow$  4 time Every iteration-

and if ( $a[i] > a[i+1]$ )

$$t = \alpha(J)$$

$$a[f] = a[f+1];$$

$$a[i+1] = t;$$

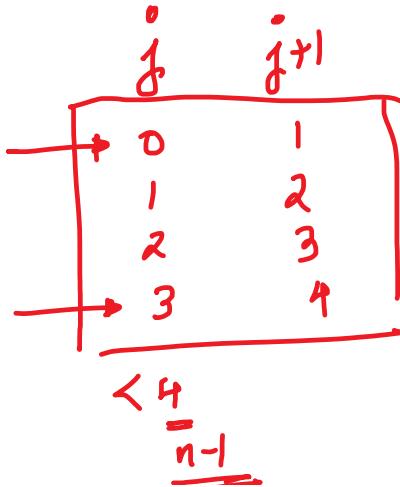
2.

2

۷

$$0 = 5 -$$

0	1	2	3	4
$\frac{20}{25}$	$\frac{25}{20}$	$\frac{10}{25}$	$\frac{5}{25}$	$\frac{25}{5}$



$$\begin{array}{r} 4 \\ \times 2 \\ \hline 8 \end{array}$$

$\rightarrow 384$

$n-1$

# C Program for Bubble Sort (Ascending Order)

## Type 1) Without Function

```
#include<stdio.h>
```

```
void main()
```

```
{
```

```
int a[25], i, t, j, n;
```

*↳ Created array of size 25*

```
printf("enter no of elements\n"); // Read value of n
scanf("%d", &n);
```

```
printf("enter the elements in array\n"); // Read n elements from user
in array
```

```
for(i=0; i<n; i++)
```

```
scanf("%d", &a[i]);
```

```
printf("original array\n"); → print original array
```

```
for(i=0; i<n; i++)
```

```
printf("%d\t", a[i]);
```

```
printf("\n");
```

*↳ sorting logic*

```
for(i=0; i<n-1; i++)
```

```
{
```

```
    for(j=0; j<n-1; j++)
```

```
{
```

*? if any element > element to its right*

```
{
```

```
        t=a[j];
        a[j]=a[j+1];
```

*Then*

*Swap*

```
t=a[j];  
a[j]=a[j+1];  
a[j+1]=t;  
}  
}
```

Swap

```
printf("\nsorted array\n"); → Print Sorted array  
for(i=0;i<n;i++)  
printf("%d\t",a[i]);  
  
printf("\n");  
return 0;  
}
```

## C Program to Sort Array Using Function - (Pointers (hd))

```
#include<stdio.h>
```

pointer (int \*a)

```
void read(int a[], int n)
{
    int i;

    for(i=0;i<n;i++)
        scanf("%d", &a[i]);
}
```

F<sup>n</sup> will read n element  
in array

```
void display(int a[], int n)
{
    int i;

    for(i=0;i<n;i++)
        printf("%d\t", a[i]);

    printf("\n");
}
```

pointer (int \*a)

⇒ Display n elements in array

```
void bubblesort(int a[], int n)
{
    int i, j, t;
    for(i=0; i<n-1; i++)
    {
        for(j=0; j<n-1; j++)
        {
            if(a[j]>a[j+1])
            {
                t=a[j];
                a[j]=a[j+1];
                a[j+1]=t;
            }
        }
    }
}
```

pointer (int \*a)

Sort an array of n elements  
in Ascending order.

```
        }  
    }  
}
```

```
int main()  
{  
int a[25],b[25],n1,n2;  
  
printf("enter no of elements in array a\n");  
scanf("%d",&n1);  
  
printf("enter no of elements in array b\n");  
scanf("%d",&n2);  
  
printf("enter the elements in aray a\n");  
  
read(a,n1);// call to read function  
    address  
printf("enter the elements in aray b\n");  
  
read(b,n2);// call to read function  
    address  
printf("original array a\n");  
  
display(a,n1);  
    address  
printf("original array b\n");  
  
display(b,n2);  
    address   ^ value of n2  
  
//sorting logic  
  
bubblesort(a,n1);  
bubblesort(b,n2);  
  
  
printf("\nsorted array a\n");  
display(a,n1);
```

```
printf("\nsorted array b\n");
display(b,n2);
```

```
return 0;
}
```

## Q) Write Program to Search an Element in Array

### (Linear Search)

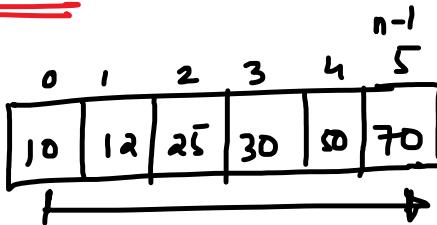
Logic  $\rightarrow$

$x = 50$

- $i=0$       is  $a[0] == x$  No
- $i=1$       is  $a[1] == x$  No
- $i=2$       is  $a[2] == x$  No
- $i=3$       is  $a[3] == x$  No
- $i=4$       is  $a[4] == x$  Yes (break)

return value of  $i$

Means  $X$  is present at  $\underline{\text{index pos}} = \underline{i}$ .



$\underline{n=6}$

$x \Rightarrow$  element to search.

$x \Rightarrow SD$

$\underline{x=15}$

- $i=0$        $a[0] == x$  No
- $i=1$        $a[1] == x$  No
- $i=2$        $a[2] == x$  No
- $i=3$        $a[3] == x$  No
- $i=4$        $a[4] == x$  No
- $i=5$        $a[5] == x$  No

$\underline{i=6}$

Since  $\underline{i == n}$   
ie  $\underline{i > n-1}$

Then  $X$  is not present in array

Without F  $\Rightarrow$  (No pointer)

```
#include<stdio.h>
int main()
{
    int a[25];
    int n,x,i;

    printf("enter no of elements\n");
}
```

```
scanf("%d",&n);

printf("enter the elements in array\n");

for(i=0;i<n;i++)
scanf("%d",&a[i]);

printf("enter element to search\n");
scanf("%d",&x);

for(i=0;i<n;i++)
{
if(a[i]==x)
break;
}

if(i==n)
printf("element %d is not present\n",x);
else
printf("element %d is found at %d index position
\n",x,i);

return 0;
}
```

## Q) Linear Search with function :-

```
#include<stdio.h>
void read(int a[],int n)
{
    int i;

    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
}

void display(int a[],int n)
{
    int i;

    for(i=0;i<n;i++)
        printf("%d\t",a[i]);

    printf("\n");
}

int LinearSearch(int a[],int n,int x)
{
    int i;
    for(i=0;i<n;i++)

```

```
{  
    if(x==a[i])  
        break;  
}  
return i;  
}  
int main()  
{  
    int a[30],n,x,p;  
  
    printf("enter nos of elements\n");  
    scanf("%d",&n);  
  
    printf("enter elements in array\n");  
    read(a,n);  
  
    printf(" elements in array are\n");  
    display(a,n);  
  
    printf("enter the element to search\n");  
    scanf("%d",&x);  
  
    p=LinearSearch(a,n,x);  
  
    if(p==n)  
        printf("element %d is not present\n",x);  
    else  
        printf("element %d is present at %d index position  
\n",x,p);
```

```
    return 0;  
}
```

Q) WAP Program to calculate mean, variance & standard deviation of array elements (without f^n).

Let

0	1	2	3	4
10	20	30	40	50



$$\begin{aligned}
 n &= 5 \\
 \text{mean} &= \frac{(a[0] + a[1] + a[2] + a[3] + a[4])}{n} \Rightarrow \left\{ \begin{array}{l} \text{for } (i=0; i < n; i++) \\ \text{mean} = \text{mean} + a[i]; \end{array} \right. + \text{sum Here} \\
 &= \frac{10 + 20 + 30 + 40 + 50}{5} = 150 \\
 \boxed{\text{mean} = (\text{mean}/5) = 30} &\quad \rightarrow \text{mean} = \underline{\underline{\text{mean}/n}};
 \end{aligned}$$

$$\begin{aligned}
 \text{Variance} &\rightarrow = \frac{1}{n} \sum_{i=0}^{n-1} (m - a[i])^2 \Rightarrow (m - a[0])^2 + (m - a[1])^2 + (m - a[2])^2 \\
 &\quad + (m - a[3])^2 + (m - a[4])^2 \\
 &\quad = (30 - 10)^2 + (30 - 20)^2 + (30 - 30)^2 + (30 - 40)^2 \\
 &\quad + (30 - 50)^2
 \end{aligned}$$

$$\begin{aligned}
 v &= 0; \\
 \text{for}(i=0; i < n; i++) &\\
 v &= v + (m - a[i]) * (m - a[i]) \\
 v &= v/n;
 \end{aligned}
 \quad \left\{ \begin{array}{l} \text{variance} = \frac{1}{5} (1000) = 200 \end{array} \right.$$

$$\text{Standard deviation} = \sqrt{\text{variance}} = \sqrt{200} = 10 \times \sqrt{2} = \underline{\underline{14.14}}$$

```
#include<stdio.h>
#include<math.h>
int main()
{
    int a[25];
    int n,i;
    float m=0,v=0,sd;
```

$$\begin{aligned}
 m &= \frac{1}{n} \sum_{i=0}^{n-1} a[i]; \\
 v &= \frac{1}{n} \sum_{i=0}^{n-1} (m - a[i])^2 \\
 sd &= \sqrt{v}
 \end{aligned}$$

formula

float m=0,v=0,sd;

*SUM = N<sup>2</sup>*

printf("enter the number of elements in array\n");  
scanf("%d",&n);

printf("enter the elements in array\n");

```
for(i=0;i<n;i++)  
{  
    scanf("%d",&a[i]);  
    m=m+a[i];  
}
```

m=m/n;

```
for(i=0;i<n;i++)  
v=v+(m-a[i])*(m-a[i));
```

v=v/n;

sd=sqrt(v);

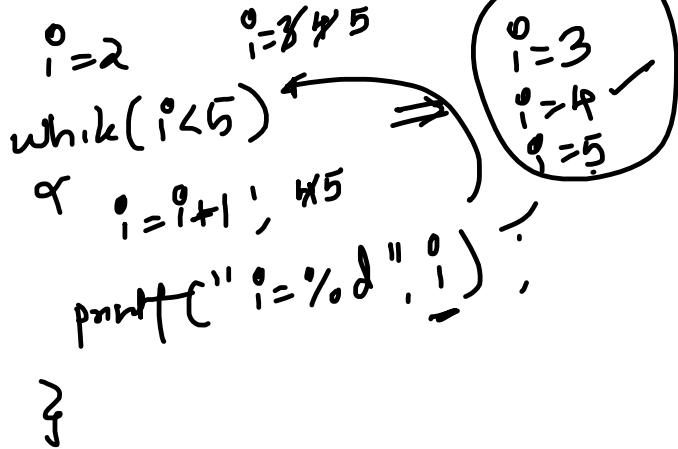
```
printf("mean = %f\n",m);  
printf("variance= %f\n",v);  
printf("standard deviation = %f\n",sd);  
return 0;  
}
```

01) ~~while~~ while ( $i < 5$ )

$\alpha \quad i = i - 1; \Rightarrow i = i + 1$   
 $\text{printf}("i = \%d", i);$

{

⇒



02)

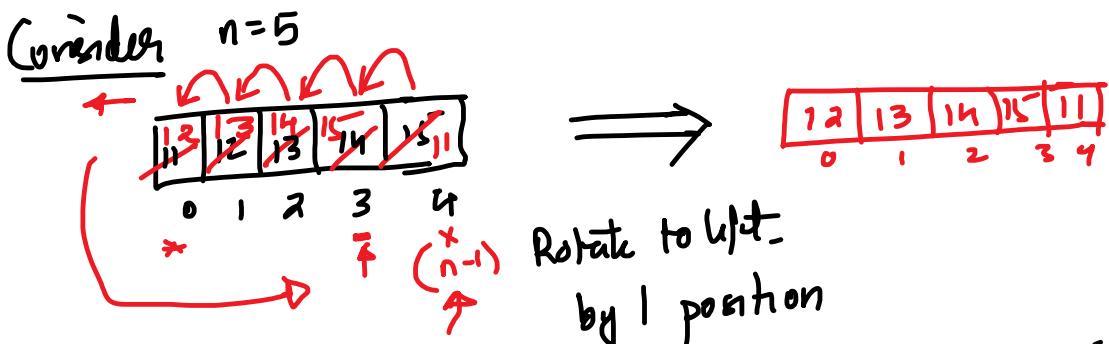
$i = 1, j = 2$

$n = i, j + 1$

$(n = i, j + 1)$



Q) WAP Program to rotate 1D array to left by 'p' position



$\text{for } (i=1; i<=p; i++) \rightarrow$  Repeat the whole logic for p time

rotate array to left by 1 position

```

1   x = a[0], // leftmost
    for(j=0; j < n-1; j++)
      { a[j] = a[j+1];
      }
    a[n-1] = x;
}

```

$j=0$   
 $j=1$   
 $j=2$   
 $j=3$   
 $j=4 *$

```

#include<stdio.h>
// function declaration
void read(int [],int); // void accept(int*, int);
void display(int [],int); // void display(int *, int);
void rotateleft(int [],int,int); // void rotateleft(int*, int, int);

int main()
{
int a[25];
int n,p;

printf("enter the number of elements in array\n");
scanf("%d",&n);

```

```
printf("enter the elements in array\n");
read(a,n);

printf("original array\n");
display(a,n);

printf("enter number of position to rotate to left\n");
scanf("%d",&p);

rotateleft(a,n,p);

printf("rotated array\n");
display(a,n);

return 0;
}

void read(int a[],int n)
{
int i;
for(i=0;i<n;i++)
scanf("%d",&a[i]);
}

void display(int a[],int n)
{
int i;
for(i=0;i<n;i++)
printf("%d\t",a[i]);

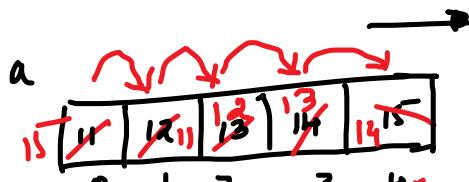
printf("\n");
}

void rotateleft(int a[],int n,int pos)
{
int i,j,x;

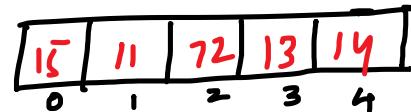
for(i=1;i<=pos;i++)// iterate the below code pos times
{
```

```
// below code will rotate array to left by 1 position  
x=a[0];// left most element  
  
for(j=0;j<n-1;j++)  
a[j]=a[j+1];  
  
a[j]=x;// x assigned to rightmost position  
  
}  
}
```

# O>WAC Program to Rotate array to right by "p" position



Rotate 1  
position  
to right



```
for(i=1; i<=p; i++)
    x = a[n-1]; // Right most
```

```
for(j=n-1; j>0; j--)
    a[j] = a[j-1];
}
```

```
a[0] = x;
```

$$x = a[n-1] = 15$$

$j = k$	$a[j] = a[j-1]$
$j = 3$	$4 \quad 3$
$j = 2$	$3 \quad 2$
$j = 1$	$2 \quad 1$
$j = 0$	$1 \quad 0$

```
#include<stdio.h>
```

```
void read(int [], int);
void display(int [], int);
void rotateright(int [], int, int);
```

```
int main()
{
    int a[25];
    int n, p;
```

```
printf("enter the number of elements in array\n");
scanf("%d", &n);
```

```
printf("enter the elements in array\n");
read(a,n);

printf("original array\n");
display(a,n);

printf("enter number of position to rotate to left\n");
scanf("%d",&p);

rotateright(a,n,p);

printf("rotated array\n");
display(a,n);

return 0;
}
```

```
void read(int a[],int n)
{int i;
for(i=0;i<n;i++)
scanf("%d",&a[i]);
}
```

```
void display(int a[],int n)
{int i;
for(i=0;i<n;i++)
printf("%d ",a[i]);

printf("\n");
}
```

```
void rotateright(int a[],int n,int pos)
{int i,j,x;

for(i=1;i<=pos;i++)
{
    x=a[n-1];// right most element
```

```
for(j=n-1;j>0;j--)  
a[j]=a[j-1];  
  
a[j]=x;  
}  
}
```

## Dynamic Memory Allocation in C

Consider

`int a[20];` Static / Compile time Memory Allocation  
 → Here at compile time (before execution) the compiler is aware of array size to create if we decides not to use exactly 20 elements but instead only 'n' element whose value will be known during execution

Here there is catch

if  $n > 25 \Rightarrow$  there is no enough memory

if  $n < 25 \Rightarrow$  There will be wastage of memory

We desire to create array of size 'n' whose value will be known as runtime known we want over the array to be created at run time.

## Runtime Memory Allocation Operator (stdlib)

① (type \*) malloc (size in bytes)  
 ↗ return address  
of specified type of memory

no of bytes to allocate

It will allocate specified no of bytes of memory  
& will return its base address.

Step 1 →  
~~\*~~ Create a pointer of  
type as same as that  
of desired

→ ~~int \*a;~~

Step 2 → Read no of elements  
desired in array

→ `scanf("%d", &n);`

Step 3 → Create memory for  
n no of elements and  
assign its base address  
to the pointer

→ ~~a = (int\*) malloc (n \* sizeof(int))~~  
~~n \* 2~~  
~~n=5 ⇒ 10 bytes~~

→ To accept Element

`for(i=0; i<n; i++)  
scanf("%d", &a[i]);`

To display

`for(i=0; i<n; i++)  
printf("%d\n", a[i]);`

```
#include<stdio.h>  
#include<stdlib.h>  
  
int main()  
{  
    int *a;  
    int n,i;  
  
    printf(" kitne elements chahiye\n");  
    scanf("%d", &n);
```

} In every earlier program

int a[25];

Replace this with

int \*a;  
scanf("%d", &n);

```
printf(" kitne elements chahiye\n");
scanf("%d",&n);

//create ana array of n integer dynamically
a=(int*)malloc(n*sizeof(int));

printf(" enter elements in array\n");

for(i=0;i<n;i++)
scanf("%d",&a[i]);

printf("\n");
return 0;
}
```

...  
scanf("%d", &n);  
a = (int\*) malloc(sizeof(int)\*n);

## Runtime Memory Allocation

②  $(\text{type} *) \text{calloc} (\text{no of elements}, \text{size of each element});$   
 OR  $(\text{type} *) \text{calloc} (\text{size of each element}, \text{no of elements});$

$\uparrow$   
 continuous allocation  
 (it's preferred while creating array dynamically)

Step 1 Create a pointer  $\rightarrow \text{int} *a;$

Step 2 > Read no of element  $\rightarrow \text{scanf}("%d", \&n);$

Step 3 Assign memory at Runtime and return its base address

$a = (\text{int} *) \underline{\underline{\text{malloc}}} (\underline{n} \times \underline{\underline{\text{sizeof(int)}}})$   
 $\uparrow$   
 address of  
 type integer

OR

$a = (\text{int} *) \text{calloc} (\underline{n}, \text{sizeof(int)});$

OR

$a = (\text{int} *) \text{calloc} (\text{sizeof(int)}, \underline{n});$

```
#include<stdio.h>
#include<stdlib.h>

int main()
{
    int *a;
    int n,i;

    printf(" kitne elements chahiye\n");
    scanf("%d",&n);

    //create ana array of n integer dynamically
    a=(int*)calloc(n,sizeof(int));

    // or   a=(int*)calloc(sizeof(int),n);

    // a=(int*)malloc(n*sizeof(int));
    // printf(" enter elements in array\n");

    for(i=0;i<n;i++)
        scanf("%d",&a[i]);

    printf(" elements in array are\n ");
    for(i=0;i<n;i++)
        printf("%d\t",a[i]);

    printf("\n");
    return 0;
}
```

Pointers To pointer  $\rightarrow$  pointer ~~is~~ pointing to another but of some type

int  $x = 10;$

int \*p;  $\rightarrow$  integer pointer  
(will store add of int type memory)

int \*\*q;  $\rightarrow$  pointer to integer pointer  
(it will store address of an integer pointer)  $\Rightarrow$

$p = \&x;$  ✓

$q = \&x;$  ✗ Error  $\rightarrow q$  can store add of pointer of type int

$q = \&p;$  ✓

printf("%x",  $\underline{\underline{x}}$ );  $\Rightarrow$  2020

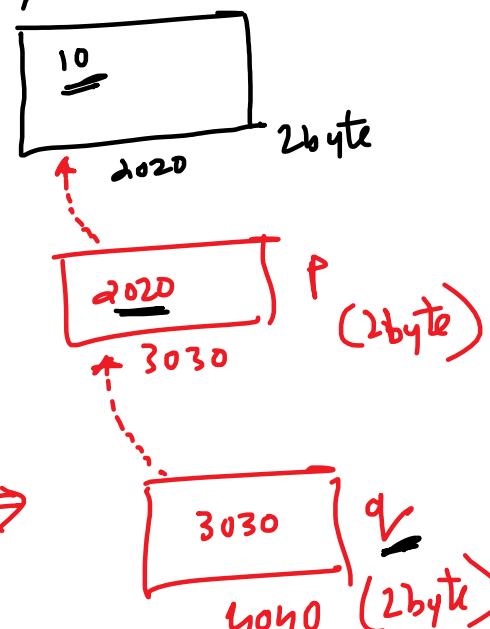
printf("%d",  $\underline{x}$ );  $\Rightarrow$  x

printf("%x",  $\underline{P}$ );  $\Rightarrow$  1020

printf("%x",  $\underline{\underline{P}}$ );  $\Rightarrow$  3030

printf(">d",  $\underline{\underline{P}}$ );  $\Rightarrow *(\underline{P}) \Rightarrow *(2020) \Rightarrow 10$   
P jisko point karta hai wala value

printf("%x",  $\underline{q}$ );  $\Rightarrow$  3030  
q, jisko point karta haun koi address



`printf("%d", &q);`  $\Rightarrow$  hold

`printf("%d", *q);`  $\xrightarrow{\text{q jisko point karta hua wala value}}$  2020

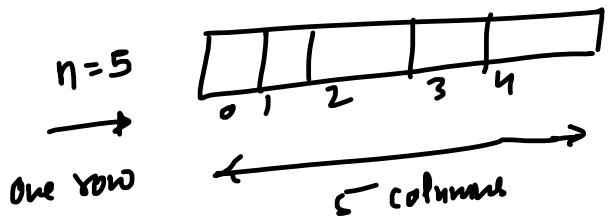
`printf("%d", *q);`  $\Rightarrow \underline{\underline{10}}$

$*(\ast q)$   
 $*(\underline{\underline{2020}})$

$\Rightarrow 10$

2D Array →

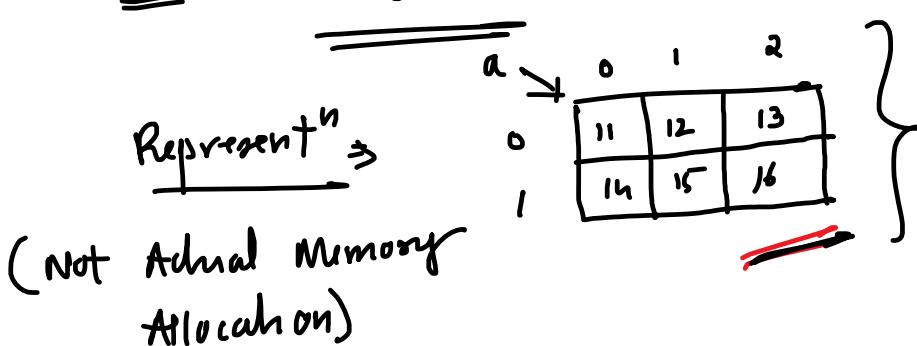
In 1D Array ⇒ only one dimension → Single row & n col



2D Array + 2 Dimension →  $\begin{matrix} \text{rows} \\ \text{col} \end{matrix}$

Declare → type arrayname [row][col],

Ex     $\text{int } a[2][3];$



Actual Memory Allocation → In actual, memory is allocated for 1D array only we make 1D array to behave like 2D array

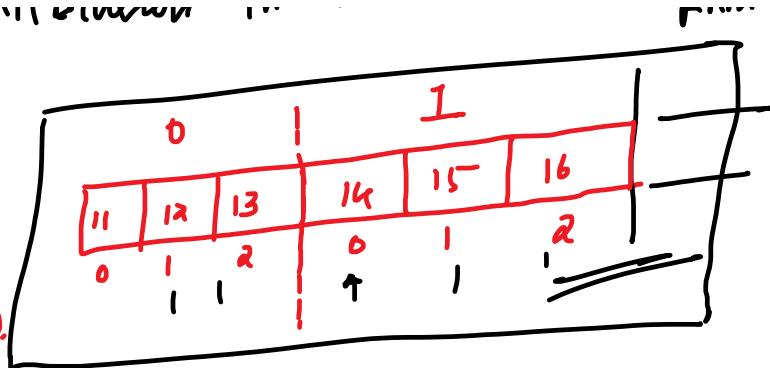
$\text{int } a[2][3];$ ,  
→ 6 element

Actual Memory Allocation in RAM

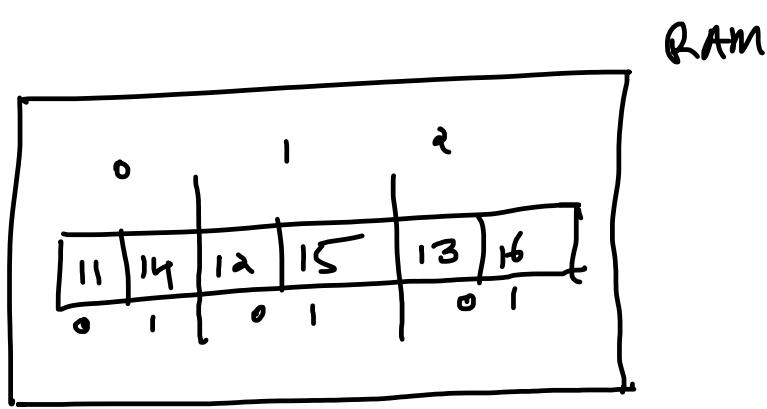


## Actual Memory Allocation

Row Major Order  
 (Row Wise Allocation)  
 (Default)

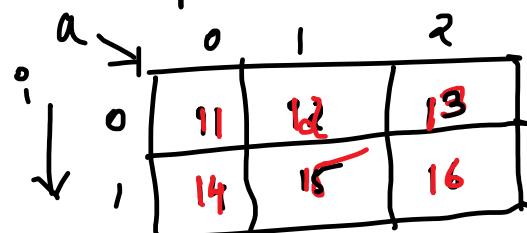


Column Major Order  
 (Col Wise Allocation)  
 (Explicitly Specified)



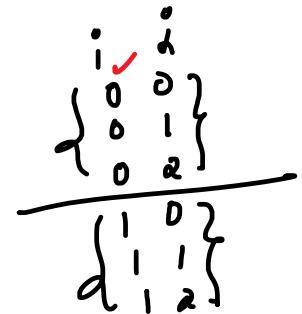
To Read Elements in 2D Array

```
int a[2][3];  
int i, j;
```



```
// Row major wise  
printf("Enter elements in 2D array\n");
```

```
for(i=0; i<rows; i++)
```



```
1  
for(j=0; j<3; j++)
```

```
1  
scanf("%d", &a[i][j]);
```

- ?



$i = 0 \rightarrow \&a[0][0]$   
 $i = 1 \rightarrow \&a[0][1]$   
 $i = 2 \rightarrow \&a[0][2]$



$\Rightarrow$  To Display Content of 2D Array

```
// Row Major Order
printf("Elements in 2D array are \n");
for (i=0; i< rows; i++)
```

	0	1	2	3
0	11	12	13	14
1	15	16	17	18
2	19	20	21	22

```
for (i=0; i< rows; i++)
```

$i \Rightarrow 0 \rightarrow 11 - 12 - 13 - 14$   
 $i \Rightarrow 1 \rightarrow 15 - 16 - 17 - 18$

}

```
for (j=0; j< cols; j++)
printf("%d ", a[i][j]);
```

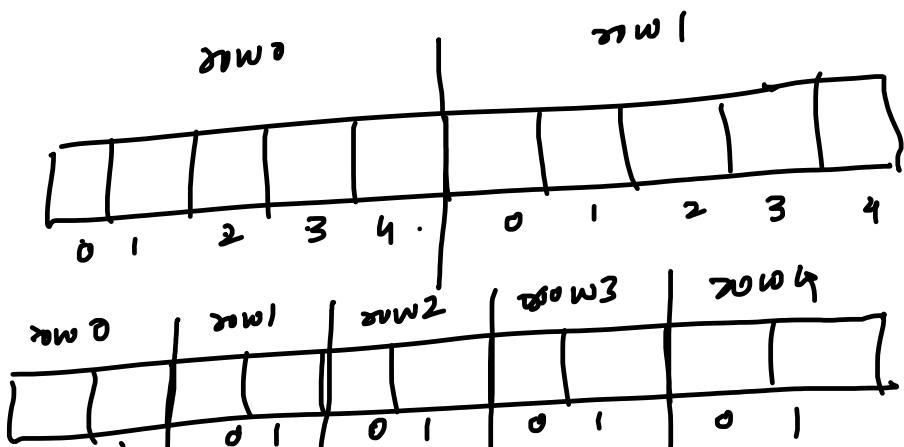
print all col of a row

printf("\n");  $\rightarrow$  take cursor to next line to print new row

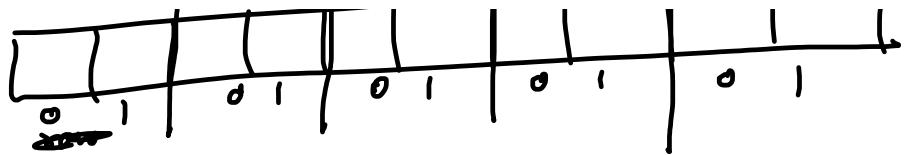
}

int a[2][5],

int a[5][2],



`int a[6][2];`



# Reading & Writing Content of 2D array without function

```
#include<stdio.h>
// without using function
int main()
{
    int a[10][10];
    int r,c,i,j;

    printf(" enter the order of 2-d array\n");
    scanf("%d %d",&r,&c);

    printf(" enter the elements in 2-d array\n");

    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
            scanf("%d",&a[i][j]);
    }

    printf("the elements in 2-d array are\n");

    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
            printf("%d ",a[i][j]);

        printf("\n");
    }

    return 0;
}
```

Q) reading & writing content in 2D array with function

```
#include<stdio.h>
// using function
void read(int[10][10],int,int);
void display(int[10][10],int,int);

int main()
{
    int a[10][10];
    int r,c,i,j;

    printf(" enter the order of 2-d array\n");
    scanf("%d %d",&r,&c);

    printf(" enter the elements in 2-d array\n");

    read(a,r,c);

    printf("the elements in 2-d array are\n");

    display(a,r,c);

    return 0;
}

void read(int a[10][10],int r,int c)// row=10 not mandatory but col=10
mandatory
{
    int i,j;
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
            scanf("%d",&a[i][j]);
    }
}
```

```
}
```

```
void display(int a[10][10],int r,int c)
{
    int i,j;
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
            printf("%d ",a[i][j]);
        printf("\n");
    }
}
```

Q) Write a C program to display Sum of all rows & Col elements in a 2 D array

Eg

	0	1	2	3	
0	10	20	30	40	→ 100
1	50	60	70	80	→ 260
2	90	40	30	20	→ 170

↓    ↓    ↓    ↓

150    120    130    140

Sum of every col element

Sum of every row elements

1/ To print sum of every row

```

for(i=0;i<r;i++)
{
    sr=0;
    for(j=0;j<c;j++)
    {
        sr=sr+a[i][j];
    }
    printf(" sum of %d index row is %d\n",i,sr);
}
  
```

	0	1	2	
0	10	20	30	40
1	50	60	70	80
2	90	100	50	60

$\begin{array}{l} i=0 \quad sr=0 \\ j=0 \quad sr = 0 + a[0][0] \\ = 0 + 10 = 10 \end{array}$ 
  
 $\begin{array}{l} j=1 \quad sr = 10 + a[0][1] \\ = 10 + 20 = 30 \end{array}$ 
  
 $\begin{array}{l} j=2 \quad sr = 30 + a[0][2] = 30 + 30 \\ = 60 \end{array}$ 
  
 $\begin{array}{l} j=3 \quad sr = 60 + a[0][3] = 60 + 40 \\ = 100 \end{array}$ 
  
 $\overline{\begin{array}{l} i=1 \\ \times \end{array}}$ 
  
Sum of 0 index row = 100
  
 $i=1$

.. + .. + ..



// To print sum of Every col

```
for(j=0; j < 4; j++)
```

of sc=0,

```
for(i=0; i < 3; i++)
```

```
sc = sc + a[i][j];
```

```
printf("Sum of %d index col is %d\n", j, sc);
```

}

		0	1	2	3	
0	10	20	30	40		
1	50	60	70	80		
2	90	100	50	60		

$$j=0 \quad sc=0$$

$$i=0 \quad sc=0 + a[0][0] \Rightarrow 10$$

$$i=1 \quad sc=10 + a[1][0] \Rightarrow 10 + 50 \\ = 60$$

```
#include<stdio.h>
```

```
int main()
```

```
{
```

```
    int a[10][10];
```

```
    int r,c,i,j;
```

```
    int sr=0,sc=0;
```

```
    printf(" enter the order of 2-d array\n");
```

```
    scanf("%d %d",&r,&c);
```

```
    printf(" enter the elements in 2-d array\n");
```

```
    for(i=0;i<r;i++)
```

```
{
```

```
        for(j=0;j<c;j++)
```

```
            scanf("%d",&a[i][j]);
```

```
}
```

```
    printf("the elements in 2-d array are\n");
```

```
    for(i=0;i<r;i++)
```

```
{
```

```

for(j=0;j<c;j++)
printf("%d ",a[i][j]);

printf("\n");
}

//operation for row sum

for(i=0;i<r;i++)
{
    sr=0;
    for(j=0;j<c;j++)
    {
        sr=sr+a[i][j];
    }
    printf(" sum of %d index row is %d\n",i,sr);

    printf("\n");
}

//operation for col sum

for(j=0;j<c;j++)
{
    sc=0;
    for(i=0;i<r;i++)
    {
        sc=sc+a[i][j];
    }
    printf(" sum of %d index col is %d\n",j,sc);

    printf("\n");
}

return 0;
}

```

WAC Preq to Add and Subtract of matrix (2D Array)

$$\begin{array}{c}
 \text{sum} \\
 \begin{array}{ccc}
 a & \begin{bmatrix} 0 & 1 & 2 \\ 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} & + & b & \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 2 & 4 & 8 \end{bmatrix} & \Rightarrow & \begin{bmatrix} 0 & 1 & 2 \\ 4 & 6 & 9 \\ 6 & 9 & 14 \end{bmatrix} \\
 & 2 \times 3 & & & 2 \times 3 & & 2 \times 3 \\
 & r & c & & r & c & r & c
 \end{array}
 \end{array}$$

logic       $\text{for } (i=0; i < r; i++)$

Addition     
 
$$\begin{array}{l}
 \text{for}(j=0; j < c; j++) \\
 \text{sum}[i][j] = a[i][j] + b[i][j];
 \end{array}$$

Subtraction       $\text{for } (i=0; i < r; i++)$

Subtraction     
 
$$\begin{array}{l}
 \text{for}(j=0; j < c; j++) \\
 \text{diff}[i][j] = a[i][j] - b[i][j];
 \end{array}$$

```
#include<stdio.h>
void read(int a[10][10], int r, int c)
{
    int i, j;
    for(i=0; i < r; i++)
        for(j=0; j < c; j++)
            a[i][j] = ...;
```

```

    {
        for(j=0;j<c;j++)
            scanf("%d",&a[i][j]);
    }
}

void display(int a[10][10],int r,int c)
{
    int i,j;
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
            printf("%d ",a[i][j]);

        printf("\n");
    }
}

void matrix_sum(int a[10][10],int b[10][10],int x[10][10],int r,int c)
{
    int i,j;
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
            x[i][j]=a[i][j]+b[i][j];

        printf("\n");
    }
}

void matrix_diff(int a[10][10],int b[10][10],int x[10][10],int r,int c)
{
    int i,j;
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
            x[i][j]=a[i][j]-b[i][j];

        printf("\n");
    }
}

```

```
}

int main()
{
    int a[10][10],b[10][10],sum[10][10],diff[10][10];
    int r,c,i,j;

    printf(" enter the order of matrix\n");
    scanf("%d %d",&r,&c);

    printf(" enter the elements in matrix a\n");

    read(a,r,c);

    printf(" enter the elements in matrix b\n");

    read(b,r,c);

    printf("the elements in matrix a\n");

    display(a,r,c);

    printf("the elements in matrix b\n");

    display(b,r,c);

    matrix_sum(a,b,sum,r,c);

    printf(" addition result\n");

    display(sum,r,c);

    matrix_diff(a,b,diff,r,c);

    printf(" substraction result\n");

    display(diff,r,c);

    return 0;
}
```

## Q) WAP Program to Transpose a Matrix

$$\begin{array}{l}
 \text{a} \\
 \left[ \begin{array}{cccc}
 0 & 1 & 2 & 3 \\
 1 & 11 & 12 & 13 & 14 \\
 15 & 16 & 17 & 18 \\
 19 & 20 & 21 & 22
 \end{array} \right] = b = a^T = \left[ \begin{array}{ccccc}
 0 & & & & \\
 11 & 12 & & & \\
 1 & 16 & 17 & & \\
 2 & 13 & 14 & 18 \\
 3 & 19 & 20 & 21 & 22
 \end{array} \right]
 \end{array}$$

$\xrightarrow{\text{3 rows}}$

As per matrix  $b$   $\xrightarrow{\text{no of rows in } b}$

$\text{for}(i=0; i < c; i++)$

$\quad \text{for}(j=0; j < r; j++)$   $\xrightarrow{\text{no of col in } b}$

$$\left\{ \begin{array}{l}
 \text{1} \quad b[i][j] = a[j][i]; \\
 \text{2} \quad \left[ \begin{array}{ccccc}
 0 & 0 & 0 & 0 & 0 \\
 0 & 1 & 1 & 1 & 1 \\
 0 & 2 & 2 & 2 & 2 \\
 0 & 3 & 3 & 3 & 3 \\
 1 & 0 & 0 & 0 & 0
 \end{array} \right]
 \end{array} \right.$$

$$\begin{array}{ccccc}
 0 & 1 & 2 & 3 & 4 \\
 1 & 0 & 1 & 2 & 3 \\
 0 & 0 & 1 & 2 & 3 \\
 \cdot & \cdot & \cdot & \cdot & \cdot \\
 0 & 0 & 0 & 0 & 0
 \end{array}$$

```
#include<stdio.h>
```

```

void read(int a[10][10], int r, int c)
{
    int i, j;
    for(i=0; i < r; i++)
    {
        for(j=0; j < c; j++)
        {

            scanf("%d", &a[i][j]);
        }
    }
}

```

```

void display(int a[10][10],int r, int c)
{
int i,j;
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{
printf("%d ",a[i][j]);
}
printf("\n");
}
}

void transpose(int a[10][10],int t[10][10],int r,int c)
{int i,j;
for(i=0;i<c;i++)// t will have c no of rows
{
for(j=0;j<r;j++)// t will have r no of cols
t[i][j]=a[j][i];
}
}

int main()
{
int a[10][10],t[10][10],r,c;

//original a will have r rows and c col
// transpose t will have c rows and r cols
printf("Enter the number of rows and cols\n");
scanf("%d%d",&r,&c);

printf("enter elements in original matrix\n");
read(a,r,c);

transpose(a,t,r,c);

printf("original matrix is \n");

```

```
display(a,r,c);

printf("transpose matrix is \n");
display(t,c,r);

return 0;
}
```

# Q) WACP to Perform Matrix Multiplication. (Imp)

Diagram illustrating matrix multiplication:

$$C = A \times B$$

Matrix A (3x3):

$$\begin{bmatrix} 0 & 1 & 2 \\ 1 & 2 & 3 \\ 2 & 3 & 0 \end{bmatrix}$$

Matrix B (3x4):

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 2 & 1 & 0 \\ 2 & 3 & 2 & 1 \\ 3 & 2 & 1 & 0 \end{bmatrix}$$

Result Matrix C (3x4):

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 2 & 1 & 0 \\ 2 & 3 & 2 & 1 \\ 3 & 2 & 1 & 0 \end{bmatrix}$$

Annotations:

- C** is circled in red.
- i** and **j** indices are shown below the matrices.
- r1 \* c1** and **r2 \* c2** are circled in red.
- i**, **j**, **K** are labeled with arrows pointing to their respective indices.
- (r1 == r2)** is circled in red.

Logic    for (  $i=0$  ,  $i < \underline{r1}$  ;  $i++$  )

$i$   
0      0

d    for(  $j=0$ ;  $j < \underline{c2}$ ;  $j++$  ) 1

q     $c[i][j] = 0$ ;

for(  $k=0$ ;  $K < \underline{c1}$ ;  $k++$  )

q     $c[i][j] = c[i][j] + \underline{a[i][k] * b[k][j]}$ ;

$$\begin{aligned}
 & c[0][0] = c[0][0] + a[0][0] * b[0][0] \\
 & = 0 + 2 \cancel{*} 1 \cancel{+} 2 \\
 & = 2 \\
 & c[0][1] = c[0][1] + a[0][1] * b[0][1] \\
 & = 0 + 1 \cancel{*} 1 \cancel{+} 1 \\
 & = 1 \\
 & c[0][2] = c[0][2] + a[0][2] * b[0][2] \\
 & = 0 + 2 \cancel{*} 2 \cancel{+} 2 \\
 & = 2 \\
 & c[0][3] = c[0][3] + a[0][3] * b[0][3] \\
 & = 0 + 1 \cancel{*} 0 \cancel{+} 0 \\
 & = 0
 \end{aligned}$$

```

#include<stdio.h>

void read(int a[10][10], int r, int c)
{
    int i,j;

    for(i=0;i<r;i++)
        for (j=0; j<c; j++)
            scanf("%d",&a[i][j]);
}

void display(int a[10][10], int r, int c)
{
    int i,j;

    for(i=0;i<r;i++)
    {
        for (j=0; j<c; j++)
            printf("%d ",a[i][j]);
        printf("\n");
    }
}

void multiply(int a[10][10],int b[10][10],int c[10][10],int
              r1,int c1,int r2,int c2)
{
    int i,j,k;
    for(i=0;i<r1;i++)
        for (j=0;j<c2;j++)
        {
            c[i][j]=0;
            for (k=0;k<c1;k++)
                c[i][j]=c[i][j]+a[i][k]*b[k][j];
        }
}

int main()
{
    int a[10][10],b[10][10],c[10][10],r1,c1,r2,c2;
}

```

$$3 \times 2 \quad 4 \times 3$$
~~not  
multiplyable~~

```
printf("Enter order of matrix1\n");
scanf("%d %d",&r1,&c1);

printf("Enter order of matrix2\n");
scanf("%d %d",&r2,&c2);

if(c1==r2)
{
    printf("enter elements in first matrix\n");
    read(a,r1,c1);
    printf("enter elements in second matrix\n");
    read(b,r2,c2);

    multiply(a,b,c,r1,c1,r2,c2);

    printf("elements of first matrix\n");
    display(a,r1,c1);

    printf("elements of second matrix\n");
    display(b,r2,c2);

    printf("elements of resultant matrix\n");
    display(c,r1,c2);
}

else
    printf("Multiplication not possible\n");

return 0;
}
```

## Strings in C

String in C is an array of characters.

Declare  $\Rightarrow$  char name [10];

To initialize.  $\rightarrow$

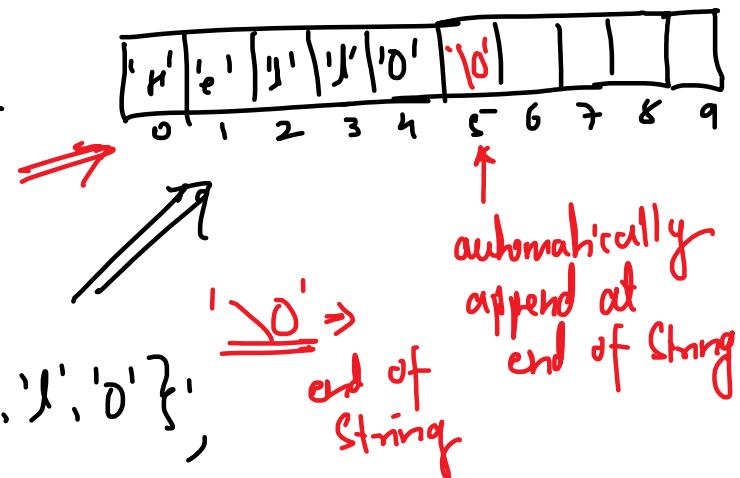


① char name [10];

```

    { name[0] = 'H';
      name[1] = 'e';
      name[2] = 'l';
      name[3] = 'l';
      name[4] = 'o'; }
```

}



② char name [10] = {'H', 'e', 'l', 'l', 'o'};

③ To Read value from User

char name[10],

printf("Enter the name\n");

scanf("%s", name);  $\rightarrow$  It will read only one word

$\uparrow$   
a complete  
String

```
#include<stdio.h>
```

```
int main()
{
    char name[15];

    printf(" enter the name\n");
    scanf("%s",name);

    printf(" Name = %s\n",name);

    return 0;
}
```

D:\ODD 2021-22\SPA INFT\SPA PROGRAMS\stringex1.exe

enter the name  
sanjeev dwivedi  
Name = sanjeev

→ only one word is read

-----  
Process exited after 4.948 seconds with return value 0  
Press any key to continue . . .

To read the complete line

gets( arrayname ) → here it reads the complete string  
( line ) and assign to the array

```
#include<stdio.h>
```

```
int main()
{
    char name[25];

    printf(" enter the name\n");
    gets(name);

    printf(" Name = %s\n",name);

    return 0;
}
```

D:\ODD 2021-22\SPA INFT\SPA PROGRAMS\stringex1.exe

enter the name  
sanjeev dwivedi  
Name = sanjeev dwivedi

→ complete line is read

-----  
Process exited after 3.674 seconds with return value 0  
Press any key to continue . . .

To read a string →

scanf("%s", name);  
gets( name );  
( one word )  
( one line )

```
    cout << name << endl;
}
return 0;
```

Summary

To point a String

10 read a String →  
gets (name);  
(one line)

printf ("%s", name);  
puts (name);

functions in <string.h> header file (Imp)  
 ↗ array of characters as input (String)

① int strlen(char \*) →

Returns length of the string passed as argument.

```
#include<stdio.h>
#include<string.h>
int main()
{
    char name[25] = "hello";
```

printf("%d\n", strlen(name));      old    5  
 return 0;  
 }

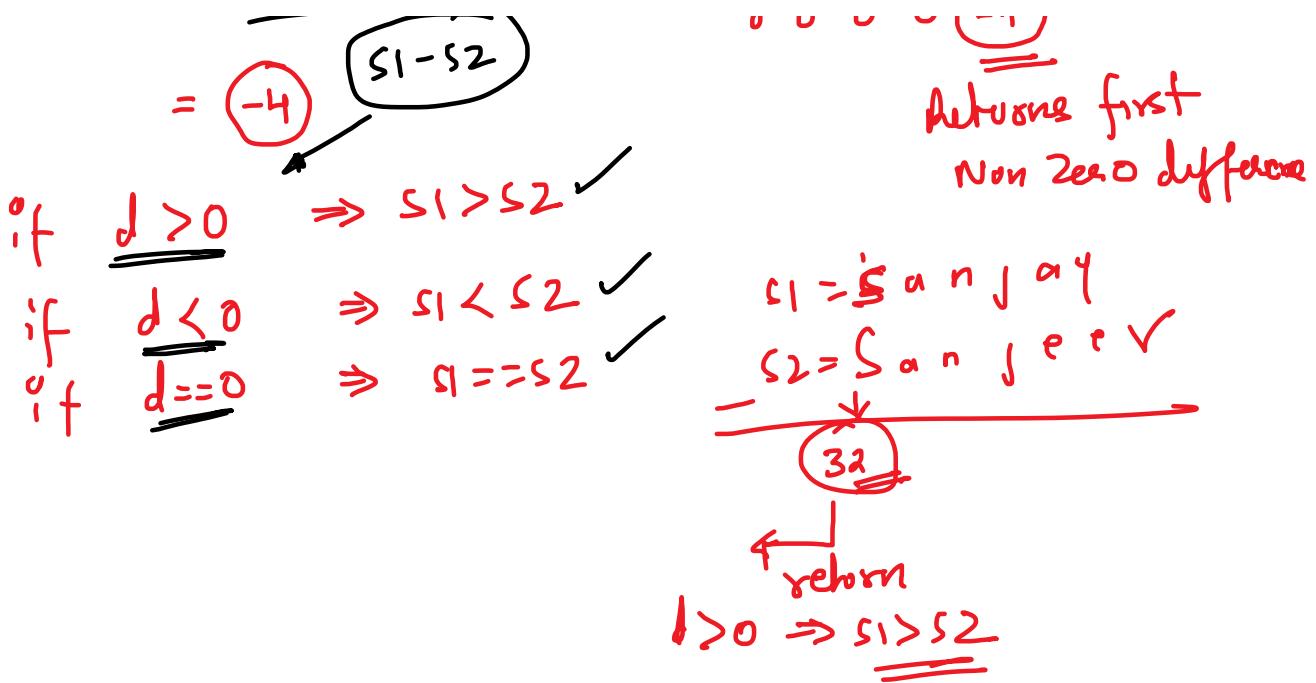
② int strcmp(char \*s1, char \*s2)

\* It returns the difference between two strings  
 passed to it.  
 → The comparison is case sensitive (comparison is based on ASCII value).  
 char s1[20] = "Sanjay";  
 char s2[20] = "Sanjeev";

int d = strcmp(s1, s2);

= (-4)

s1 = S a n j a y  
 - s2 = S a n j e e v  
 0 0 0 0 -4  
1. Line no first



ASCII Values →	
A	B
65	66
+32	
a	b
97	98
Z	
90	
-32	
38	
122	

### ③ int strcmp( char \*s1, char \*s2 )

It returns diff bet<sup>n</sup> passed String

\* Comparison is case Insensitive

$\text{char } s1[7] = \text{Sanjay}$   
 $\text{char } s2[8] = \text{Sanjeev} \checkmark$   
 $\text{strcmp} \quad - \quad \underline{\quad \quad \quad \quad \quad \quad \quad \quad \quad}$   
-4

#### ④ void strcat (char \*s1, char \*s2)

it will concatenate the string passed as second arg at the end of string passed as first arg

Ex

char s1 [15] = "Good";  
char s2 [25] = "Morning";

strcat (s1, s2);

puts(s1);  $\Rightarrow$  Good Morning

puts(s2);  $\Rightarrow$  Morning

#### ⑤ void strcpy (char \*s1, char \*s2) $\Rightarrow$ it will copy

the string passed as second arg into the string passed as first arg

char s1 [25] = "Good";

char s2 [25] = "Morning"

strcpy (s1, s2);

puts(s1);  $\Rightarrow$  Morning ✓

puts(s2);  $\Rightarrow$  Morning ✓

#### ⑥ void stored(char \*s1) $\Rightarrow$ it reverses the string passed to it as argument -

char name [25] = "Hello";

`strev(name);`  
`puts(name);`  $\Rightarrow$  olleh.

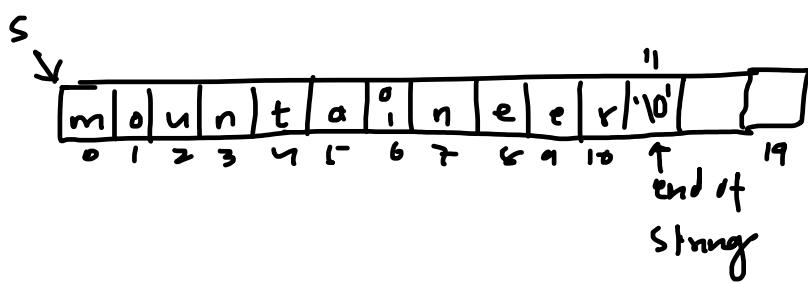
Q) WAP Program to read a string from user & Count no of vowels in it

```
#include<stdio.h>
#include<string.h>
int main()
{
char s[20];
int i,c=0;

printf("enter the string\n");
gets(s);

for(i=0;i<strlen(s);i++)
{
switch(s[i])
{
    case 'A':
    case 'a':
    case 'E':
    case 'e':
    case 'I':
    case 'i':
    case 'O':
    case 'o':
    case 'U':
    case 'u': c++;
}
}

printf("numbers of vowels in string %s is %d\n",s,c);
return 0;
}
```



c = 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

\* for (i=0 ; i< strlen(s) ; i++)

if ( s[i] == 'A' || s[i] == 'a' || s[i] == 'E' || s[i] == 'e' || s[i] == 'I' || s[i] == 'i' || s[i] == 'O' || s[i] == 'o' || s[i] == 'U' || s[i] == 'u' )

if ( s[i] == 'A' || c[i] == 'a' || c[i] == 'E' ) - - - - -  
}        c++;

Q) WAP to determine a String is palindrome or not  
(without using strrev()) ↑  
 If reverse of the String  
 is String itself.

Logic

char s[25] = "Malayalam";

m	a	l	a	y	a	l	a	m.	'\0'	---	24
0	1	2	3	4	5	6	7	8			



$$i=0;$$

$$f = \underline{\text{strlen}(s)} - 1;$$

$$\text{int } f = 1;$$

while ( $i < j$ )

{

$$\text{if } (s[i] == s[j])$$

{  $f = 0$ ; break; }

else  
 $i++;$   
 $j--;$

$$f = \boxed{1}$$

m	a	l	a	y	e	l	a	m	'\0'
0	1	2	3	4	5	6	7	8	

$$s[3] == s[5]$$

$\therefore f = 0$  break

m	a	l	a	a	l	a	m
---	---	---	---	---	---	---	---

if ( $f == 1$ )

printf("Palindrome Ha!\\n");

else

printf("Palindrome Nah!\\n");

```
#include<stdio.h>
#include<string.h>
int main()
{   char s1[25];
    int i,j,f=1;
    printf("enter the string\n");
    gets(s1);

    i=0;
    j=strlen(s1)-1;

    while(i<j)
    {
        if(s1[i]!=s1[j])
        {
            f=0;
            break;
        }
        i++;
        j--;
    }

    if(f==0)
        printf("string %s is not a palindrome\n",s1);
    else
        printf("string %s is a palindrome\n",s1);
    return 0;
}
```

Q) Write program to sort n strings in Alphabetical Order.  
(Ascending Order)

Sol →

char  $s[10][10]$ ;  
 ↑ max 10 strings   ↑ max length of every string is 10

	0	1	2	3	4	5	6	7	8	9
$s[0]$	a	j	a	y	'\0'					
$s[1]$	v	i	j	a	y	'\0'				
$s[2]$	s	a	n	j	a	y	'\0'			
$s[3]$	s	a	n	j	p	c	v	'\0'		
$s[4]$	b	a	i	j	u	'\0'	.			

BubbleSort      char  $t[20]$ ;  
 for ( $i=0$ ;  $i < n$ ;  $i++$ )

    for ( $j=0$ ;  $j < n-1$ ;  $j++$ )

        if ( $s[j] > s[j+1]$ )  $\rightarrow$  strcmp( $s[j], s[j+1]$ )  $> 0$ ,

        swap  $s[j]$  &  $s[j+1]$ ,

?      ?      ?

$\text{strcmp}(s1, s2) > 0 \Rightarrow s1 > s2$   
 $\leq 0 \Rightarrow s1 \leq s2$   
 $= 0 \Rightarrow s1 == s2$

$t = s[j]; \rightarrow \text{strcpy}(t, s[j])$   
 $s[j] = s[j+1]; \rightarrow \text{strcpy}(s[j], s[j+1])$   
 $s[j+1] = t \rightarrow \text{strcpy}(s[j+1], t);$

```
#include<stdio.h>
#include<string.h>
```

```

int main()
{
    char st[20][20];
    int i,j,n;
    char t[20];// for swapping

    printf("enter the no of strings\n");
    scanf("%d",&n);

fflush(stdin);//flush the inputstream before reading char or
string

    printf("enter the strings\n");

    for(i=0;i<n;i++)
        gets(st[i]);

    printf("original order of strings\n");
    for(i=0;i<n;i++)
        printf("%s\t",st[i]);

    printf("\n");

//bubble sort
    for(i=0;i<n-1;i++)
    {
        for(j=0;j<n-1;j++)
        { //st[j]>st[j+1]

            if(strcmp(st[j],st[j+1])>0)
            {

strcpy(t,st[j]);//t=st[j]
strcpy(st[j],st[j+1]);//st[j]=st[j+1]
strcpy(st[j+1],t);//st[j+1]=t

            }
        }
    }

printf("alphabetical order of strings\n");
for(i=0;i<n;i++)
printf("%s\t",st[i]);

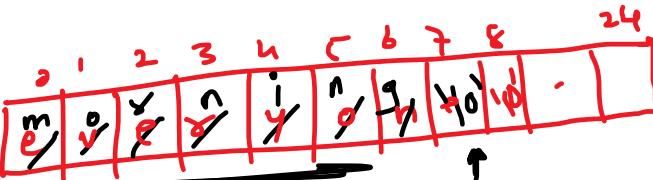
```

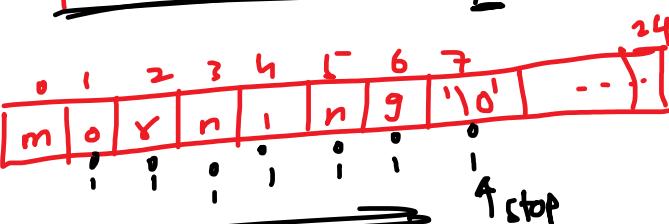
```
return 0;  
}
```

Q) WAP program to copy one string to another

Without Using strcpy

logic

char s1[25] = 

char s2[25] = 

To Copy s2  $\rightarrow$  s1

$i = 0;$

Jab tak s2 khatam  
Nahi ho jaata

while( $s2[i] \neq '\backslash 0'$ )

{  $s1[i] = s2[i];$

}  $i++;$

$s1[i] = '\backslash 0';$  // terminate s1 string

```
#include<stdio.h>
int main()
{
    char s1[25], s2[25];
    int i;

    printf("enter the first string\n");
    gets(s1);
    printf("enter the second string\n");
    gets(s2);
```

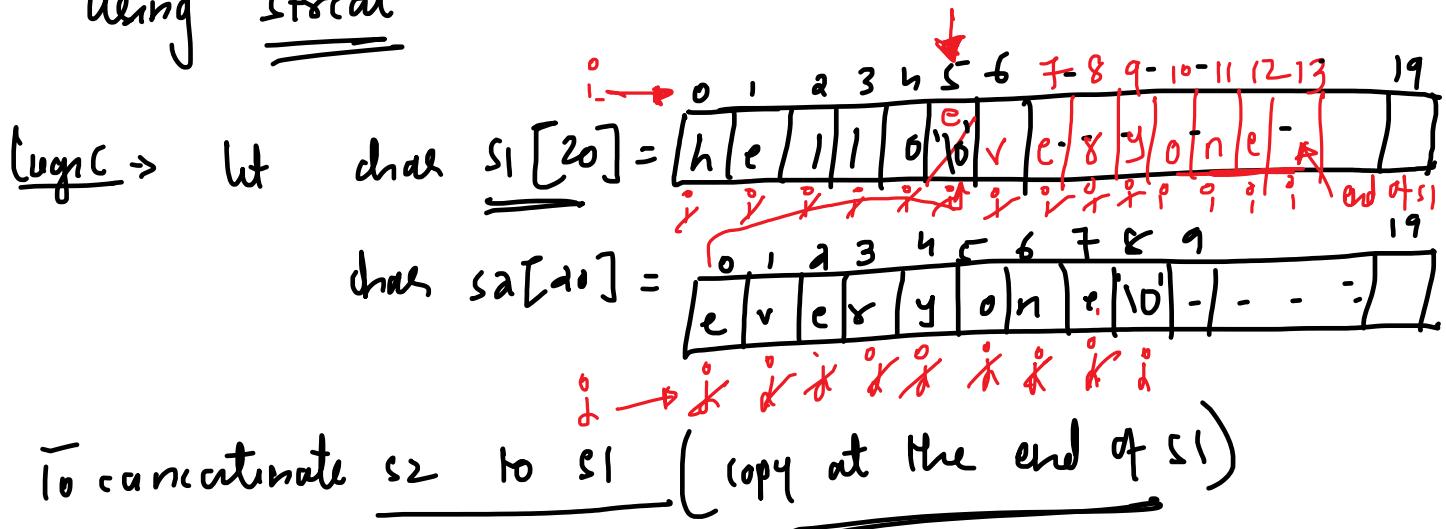
```
printf("before copy operation\n");
printf("s1= %s\n",s1);
printf("s2= %s\n",s2);

i=0;
while(s2[i]!='\0')
{
    s1[i]=s2[i];
    i++;
}
s1[i]='\0';//end explicitly

printf("after copy operation\n");
printf("s1= %s\n",s1);
printf("s2= %s\n",s2);

return 0;
}
```

Q) Write a C Program to Concatenate two Strings without using strcat



\* Let there be an i pointer that goes to end of  $s_1$ .

Step 1

```
i=0;
while(s1[i] != '\0')
    i++;
```

This will take  $i$  to  
the end of  $s_1$   
(immediately next position  
after  $s_1$ )

OR

```
i = strlen(s1);
```

\*

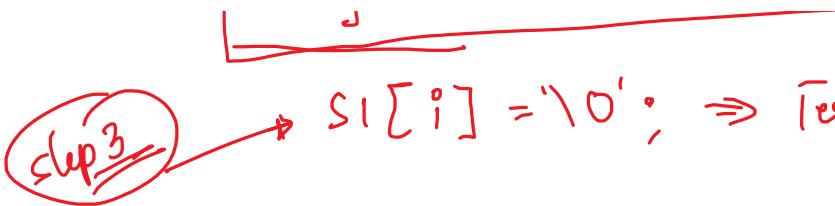
Now

```
j=0; → start of string  $s_2$ 
while(s2[j] != '\0') ⇒  $s_2$  string khatam nahi
{   s1[i] = s2[j];
    i++;
    j++;
}
```

Step 2

-

∴  $s_1$  = "Hello world" + "everyon" = "Hello world everyon"

 Step 3  
s1[i] = '\0';  $\Rightarrow$  Terminate String s1

```
#include<stdio.h>
```

```
int main()
{
char s1[25];
char s2[25];
int i,j;

printf("enter the first string\n");
gets(s1);
printf("enter the second string\n");
gets(s2);
```

```
printf("original string\n");
printf("first string = %s\n",s1);
printf("second string= %s\n",s2);
```

```
i=0;
//take i to end of first string s1
//i=strlen(s1);
while(s1[i]!='\0')
i++;
```

```
j=0;
while(s2[j]!='\0') //jab tak s2 khatam nahi ho jaata
{
    s1[i]=s2[j]; // copy s2 to s1
    i++;
    j++;
}
```

```
s1[i]='\0'; //explicit ending of first string
```

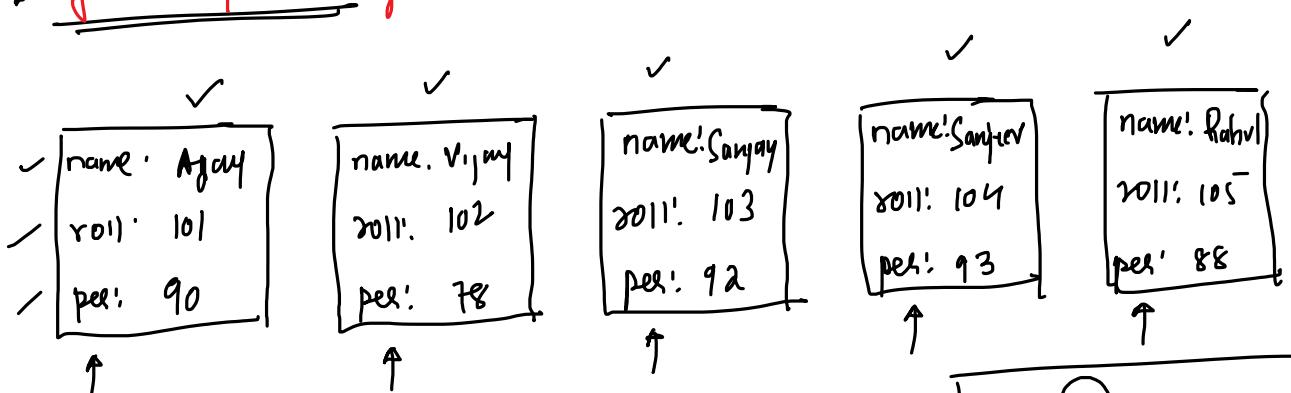
```
printf("after concat operation\n");
printf("first string = %s\n",s1);
printf("second string= %s\n",s2);

return 0;
}
```

## VImp → Structure

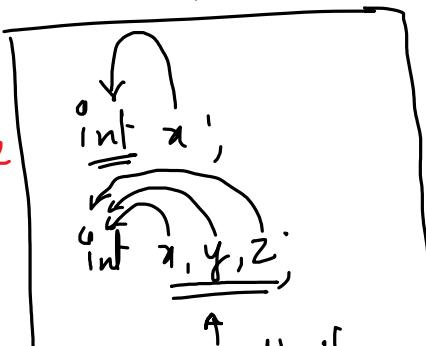
- \* All the programs written upto now were procedure oriented (Computation/Implementation of formula)
- \* Importance was "Kaise Luen a Mai"
  - i.e. Procedure was Important & not the Data.
- \* Now In Real World Scenario, Procedure is not Important, Data is Important

Consider → We have to read name, roll, per of 5 students and display name of the student with highest percentage.



User Defined Data Type →

for user defined type      keyword that specifies it is structure  
typedef      struct      student      structure name



typedef struct student

```
{
    char name[25];
    int roll;
    float per;
}
```

} ; ← End of Structure

Structure Member

Size =  $25 \times 1 = 25$  byte → name  
 2 byte → roll  
 4 byte → per

31 byte

Size of Structure

int x,y,z,  
 ↑  
 Variable of type int  
 ↑  
 type

\* We have created a data type Student such that every Student will have name, roll & per;

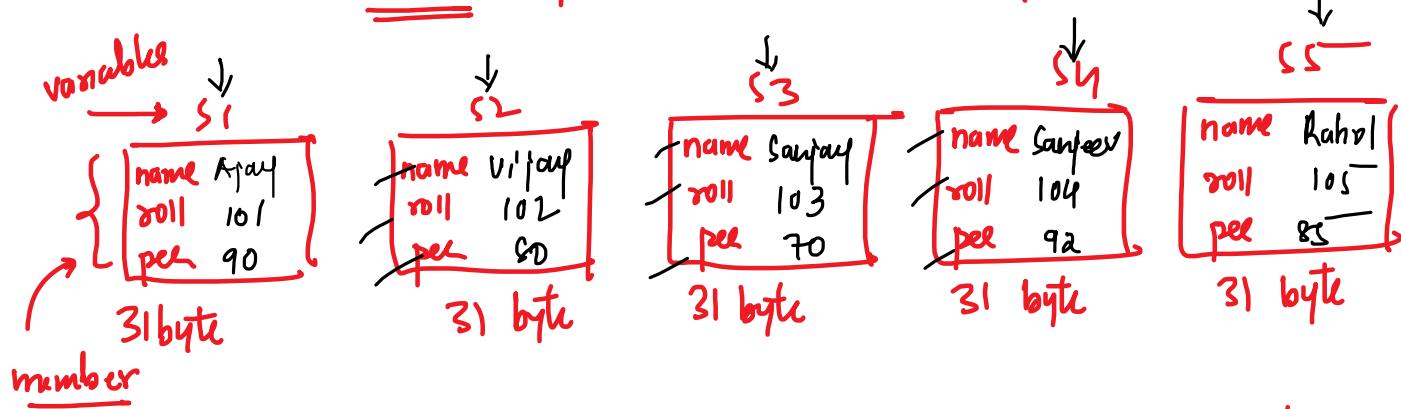
\* To realize / use a type we must Create variable of that type

Student s1,s2,s3,s4,s5;  
 (31 byte)

\* Variables of type student  
 \* Variable of type student will have all the properties (Member) of the structure

int x,y,z;  
 ↑  
 variables of type int

\* The variables will have all the properties of the type



... ↗ number of each variable.

\* To fill/Assign values to members of each variable.

```
printf("Enter details for first Student");
```

```
printf("Enter name");
```

```
scanf("%s", s1.name); // gets s1.name;
```

```
printf("Enter roll\n");
```

```
scanf("%d", &s1.roll);
```

```
printf("Enter percentage\n");
```

```
scanf("%f", &s1.per);
```

To assign value to the member we must specify the variable

```
printf("Name = %s\n", s1.name);
```

```
printf("Roll = %d\n", s1.roll);
```

```
printf("Per = %.f\n", s1.per);
```

```
printf("Enter details of Second Student\n");
```

```
printf("Enter name\n");
```

```
scanf("%s", s2.name);
```

```
printf("Enter roll\n");
```

```
scanf("%d", &s2.roll);
```

```
printf("Enter per\n");
```

```
scanf("%f", &s2.per);
```

Q) WAP Program to Create a Structure Student with data member as name, roll, kt. Accept & display details for 1 student.

```
#include<stdio.h>

typedef struct
{
    char name[20];
    int roll;
    int kt;
}Student;

int main()
{
    Student s1;

    printf("enter the details of student\n");
    printf("enter name\n");
    gets(s1.name);
    printf("enter the roll\n");
    scanf("%d",&s1.roll);
    printf("enter number of kt\n");
    scanf("%d",&s1.kt);

    printf("details of student\n");
    printf("name= %s\n",s1.name);
    printf("roll= %d\n",s1.roll);
    printf("kt= %d\n",s1.kt);

    return 0;
}
```

Q) WAP Program to Create a Structure Student with data member as name, roll & per. Accept details for 5 Students and display details of student with highest percentage.

Logic

```

typdef struct
{
    char name[20];
    int roll,
        float per,
} Student;

```

Student S[5];

Array of 5 variables of type Student

name = kavya	name = Vinita	name = Samayak	name = Rahul	name = Sanjeev
roll = 104	roll = 102	roll = 103	roll = 109	roll = 105
per = 90	per = 95	per = 80	per = 78	per = 88

0 1 2 3 4

To fill values

int i;

for (i = 0; i < 5; i++)

```

    printf("Enter Details of %d student \n", i+1);
    printf("Enter name\n");
    gets(s[i].name);
    printf("Enter roll\n");
    scanf("%d", &s[i].roll);
    printf("Enter percentage\n");
    scanf("%f", &s[i].per);
}

```

name = kavya	name = Vinita	name = Samayak	name = Rahul	name = Sanjeev
--------------	---------------	----------------	--------------	----------------

✓

name = Kajay	name = Vinitay	name = Samayay	name = Rahul ✓	name = Santeev
roll = 104	roll = 102	roll = 103	roll = 104	roll = 105
per = 90	per = 95	per = 80	per = 98	per = 88

0  
↑

1

2

3  
↑

4

max = s[0].per;  
 pos = 0

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

{

if (s[i].per > max)

{ max = s[i].per;  
 pos = i;

}

<u>Ex</u>	<u>i = 3</u>
max = 90 ✓	i = 3
pos = 0	s[3].per > max
<u>i = 1</u>	<u>max = s[3].per</u> <u>= 98</u>
"Is s[1].per > max Yes	
max = s[1].per ⇒ 95	
pos = 1	
<u>i = 2</u>	<u>i = 2</u>
"Is s[2].per > max No	
<u>i = 4</u>	<u>i = 4</u>
"Is s[4].per > max No	

// now display details of student at position pos

```
printf("Name=%s\n", s[pos].name);
printf("Roll=%d\n", s[pos].roll);
printf("Per=%f\n", s[pos].per);
```

→

```
#include<stdio.h>
```

```
typedef struct
{
char name[20];
int roll;
float per;
}Student;
```

```
int main()
{
Student s[5];
```

```
int i,p;
float max;
printf("enter details of students\n");

for(i=0;i<5;i++)
{
printf("for %d student\n",i+1);

printf("enter name\n");
gets(s[i].name);

printf("enter the roll\n");
scanf("%d",&s[i].roll);

printf("enter percentage\n");
scanf("%f",&s[i].per);

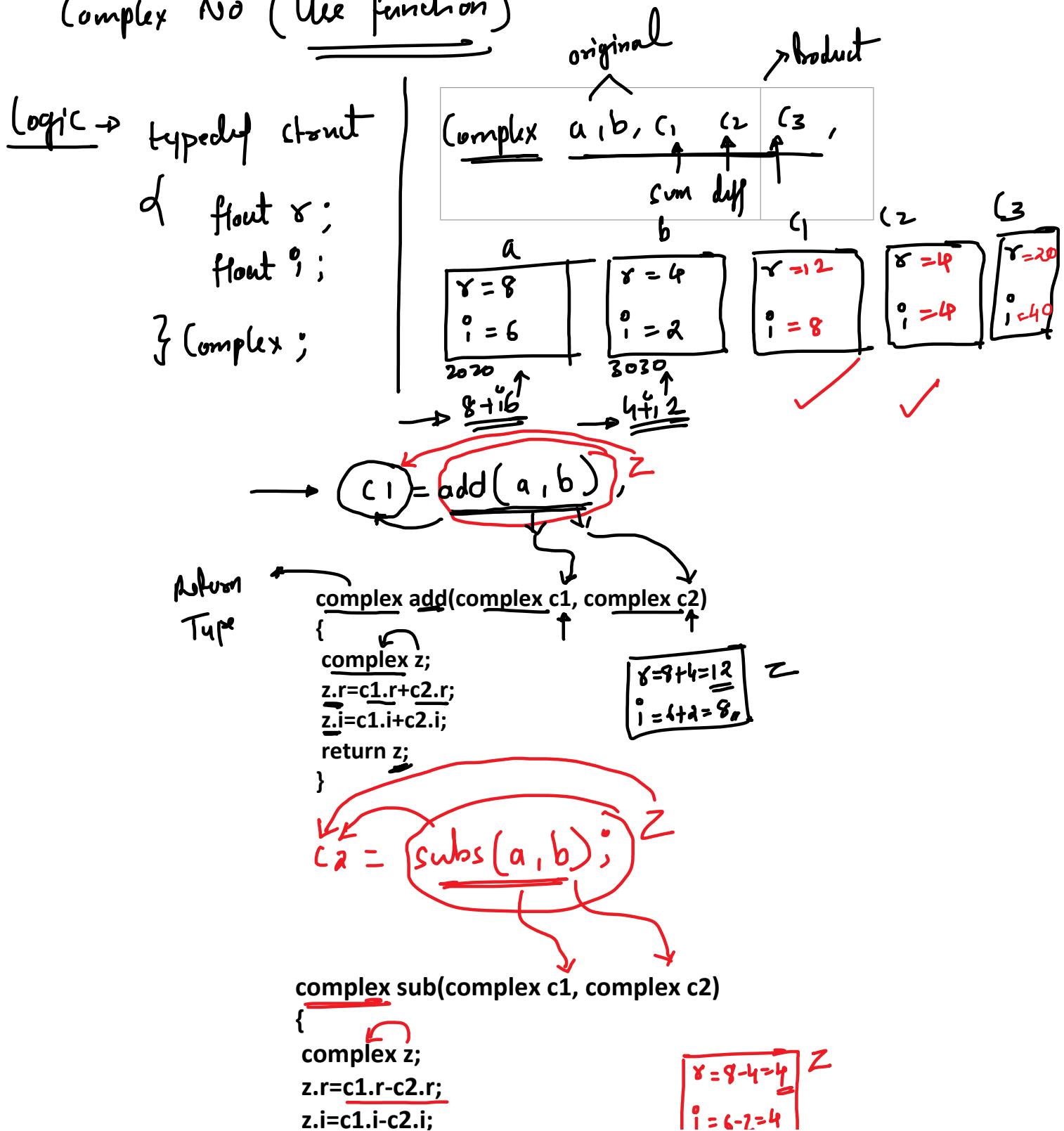
fflush(stdin);
}

max=s[0].per;
p=0;

for(i=1;i<5;i++)
{
    if(s[i].per>max)

    {
        max=s[i].per;
        p=i;
    }
}
printf("details of student with max per\n");
printf("name= %s\n",s[p].name);
printf("roll= %d\n",s[p].roll);
printf("per= %f\n",s[p].per);
return 0;
}
```

Q) Write a C program to create a structure Complex with data members as  $r$  (real) &  $i$  (imag). Perform Addition, Subtraction and Multiplication of two Complex No (Use function)

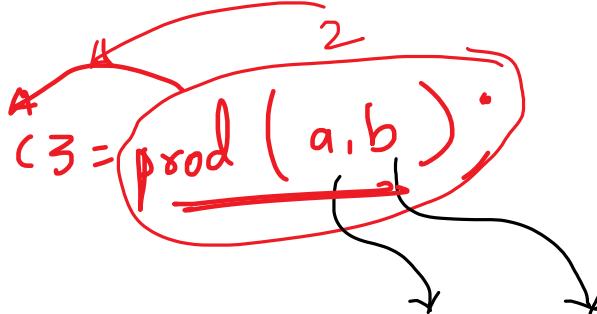


```

z.r=c1.r-c2.r;
z.i=c1.i-c2.i;
return z;
}

```

$$\begin{array}{l}
r = 8 - 4 = 4 \\
i = 6 - 2 = 4
\end{array}$$



complex prod(complex c1, complex c2)

```

{
    complex z;
    z.r=c1.r*c2.r-c1.i*c2.i;
    z.i=c1.r*c2.i+c1.i*c2.r;
    return z;
}

```

$$\begin{array}{l}
r = 8 * 4 - 6 * 2 \Rightarrow 32 - 12 \Rightarrow 20 \\
i = 8 * 2 + 6 * 4 \Rightarrow 16 + 24 \Rightarrow 40
\end{array}$$

$$(8 + 6i) \times (4 + 2i)$$

$\overset{a}{(8+6i)}$      $\overset{b}{(4+2i)}$   
 ↑                  ↑  
 $c_{1,r}$      $c_{1,i}$               ↑                  ↑  
 $c_{2,r}$      $c_{2,i}$

$$\Rightarrow (8 * 4) + (8 * 2)i + (6 * 1i * 4) - (6 * 2)$$

$$\Rightarrow \underline{(8 * 4)} - \underline{(6 * 2)} + i \left( \underline{\underline{8 * 2}} + \underline{\underline{6 * 4}} \right)$$

$\overset{\text{real}}{\underline{(8 * 4)}} \quad \overset{\text{real}}{\underline{(6 * 2)}} \quad \overset{i\text{mag}}{\underline{\underline{(8 * 2)}}} \quad \overset{i\text{mag}}{\underline{\underline{(6 * 4)}}}$   
 $\overset{\text{real}}{\underline{\underline{c_{1,r} * c_{2,r}}}} \quad \overset{\text{real}}{\underline{\underline{c_{1,i} * c_{2,i}}}} \quad \overset{i\text{mag}}{\underline{\underline{c_{1,r} * c_{2,i}}}} \quad \overset{i\text{mag}}{\underline{\underline{c_{1,i} * c_{2,r}}}}$

```
#include<stdio.h>
typedef struct
{
    float r,i;
}complex;
```

```
complex add(complex c1, complex c2)
{
    complex z;
    z.r=c1.r+c2.r;
    z.i=c1.i+c2.i;
    return z;
}
```

```
complex sub(complex c1, complex c2)
{
    complex z;
    z.r=c1.r-c2.r;
    z.i=c1.i-c2.i;
    return z;
}
```

```
complex prod(complex c1, complex c2)
{
    complex z;
    z.r=c1.r*c2.r-c1.i*c2.i;
    z.i=c1.r*c2.i+c1.i*c2.r;
    return z;
}
```

```
void display(complex t)
{
    if (t.i>=0)
        printf("%f + %f i\n",t.r,t.i);

    else printf("%f %f i\n",t.r,t.i);
}
```

```
int main()
{
    complex a,b,c1,c2,c3;

    printf("Enter complex number 1\n");
    scanf("%f%f",&a.r,&a.i);
    printf("Enter complex number 2\n");
    scanf("%f%f",&b.r,&b.i);
```

```
c1=add(a,b);
printf("Addition = ");
display(c1);
```

```
c2=sub(a,b);
printf("Subtraction = ");
display(c2);
```

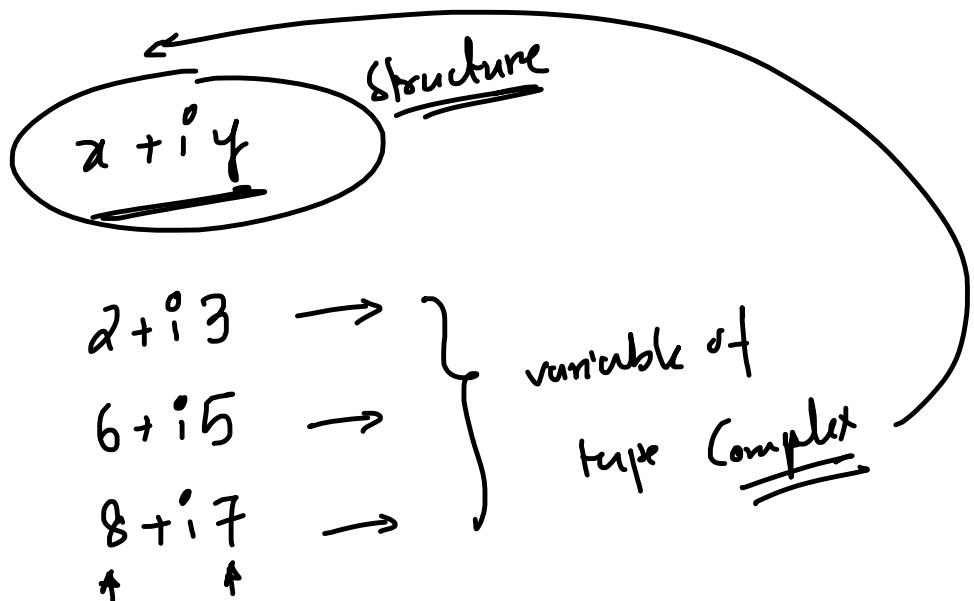
```
c3=prod(a,b);
printf("Multiplication = ");
display(c3);
```

```
return 0;
}
```

~~IA~~ Khatam  $\Rightarrow$  24 July  $\rightarrow$  Saturday CP IA  $\Rightarrow$  23<sup>rd</sup> July  
ESE  $\rightarrow$  5 August Timing  $3 \rightarrow 4$

Practical  $\Rightarrow$  26 July CP ?

Complex



Distance →

8 feet 10 inches  
11 feet 6 inches  
8 feet 4 inches

variable of  
type distance  
feet  
inches

typedef struct  
{ float feet;  
float inches;  
} Distance;

Q) Write a C program to create structure Distance with data members as feet and inches.

and add 2 distance.

```
#include <stdio.h>
```

```
typedef struct
{
    int feet;
    int inch;
} distance,
```

```
void display(distance d)
{
    printf("%d feet and %d inches\n",
           d.feet, d.inch);
}
```

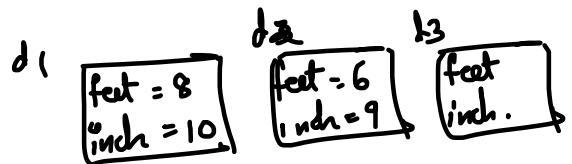
```
distance sum(distance d1, distance d2)
```

```
{  
    distance d';
```

$$d'.feet = d1.feet + d2.feet;$$

$$d'.inch = d1.inch + d2.inch;$$

```
if(d.inch > 12)
```



```
int main()
```

```
{  
    distance d1, d2, d3;
```

```
printf("Enter feet and inches for first dist\n");
scanf("%d %d", &d1.feet, &d1.inch);
```

```
printf("Enter feet and inches for Second dist\n");
scanf("%d %d", &d2.feet, &d2.inch);
```

$$d3 = \underline{\underline{sum(d1, d2)}};$$

```
printf("First distance\n");
```

```
display(d1);  $\Rightarrow$  8 feet and 10 inch.
```

```
printf("Second distance\n");
```

```
display(d2);  $\Rightarrow$  6 feet and 9 inch.
```

```
printf("Resultant distance\n");
```

```
display(d3);
```

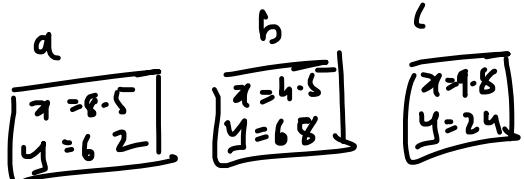
```
return 0;
```

```
if(d.inch > 12)
{ d.feet = d.feet + (d.inch/12);
  d.inch = (d.inch%12);
}
return d;
}
```

① WAP Program to Create a Structure Point (x,y) with x,y as Members Determine whether 3 points are collinear or not - (Ques)

typedef struct  
{ float x,y;  
} point;

$\Rightarrow$  Point a,b,c;



```
#include<stdio.h>
#include<math.h>
typedef struct
{
    float x,y;
}point;

float dist(point p, point q)
{return(sqrt((p.x-q.x)*(p.x-q.x)+(p.y-q.y)*(p.y-q.y)));}
}

int main()
{
    point a,b,c;
    float d1,d2,d3;

    printf("Enter x and y coordinates of point1\n");
    scanf("%f%f",&a.x,&a.y);

    printf("Enter x and y coordinates of point2\n");
    scanf("%f%f",&b.x,&b.y);

    printf("Enter x and y coordinates of point3\n");
}
```

```
scanf("%f%f",&c.x,&c.y);

d1=dist(a,b);
d2=dist(b,c);
d3=dist(c,a);

printf("%f %f %f\n",d1,d2,d3);

if (d1+d2==d3 || d2+d3==d1 || d3+d1==d2)
    printf("Collinear");

else printf("Non-collinear");

return 0;
}
```

Flowchart  $\rightarrow$  Pictorial representation of flow of control in Algorithm

Notations  $\rightarrow$



$\rightarrow$  start/End



$\rightarrow$  for I/P or O/P



$\Rightarrow$  Processing



$\Rightarrow$  Flow



$\rightarrow$  Condition.



$\rightarrow$  Same Page Connector

Flowchart for Root of Quadratic Eq<sup>n</sup>.

