① What is Algorithm?

→ An algorithm is a step-by-step procedure to perform particular task.

It should not contain error (

* Finite set of Instruction.

Imp.

② characteristics of Algorithm?

→ ① Input :-

It takes one or more inputs to operate on.

② output :-

~~It is produces zero or one Inputs one or more defined inputs, which are the value or~~

It is produce one or more well defined outputs, which are the results of its operations

③ Definiteness :-

Each step in the algorithm must be clear, and precisely defined.

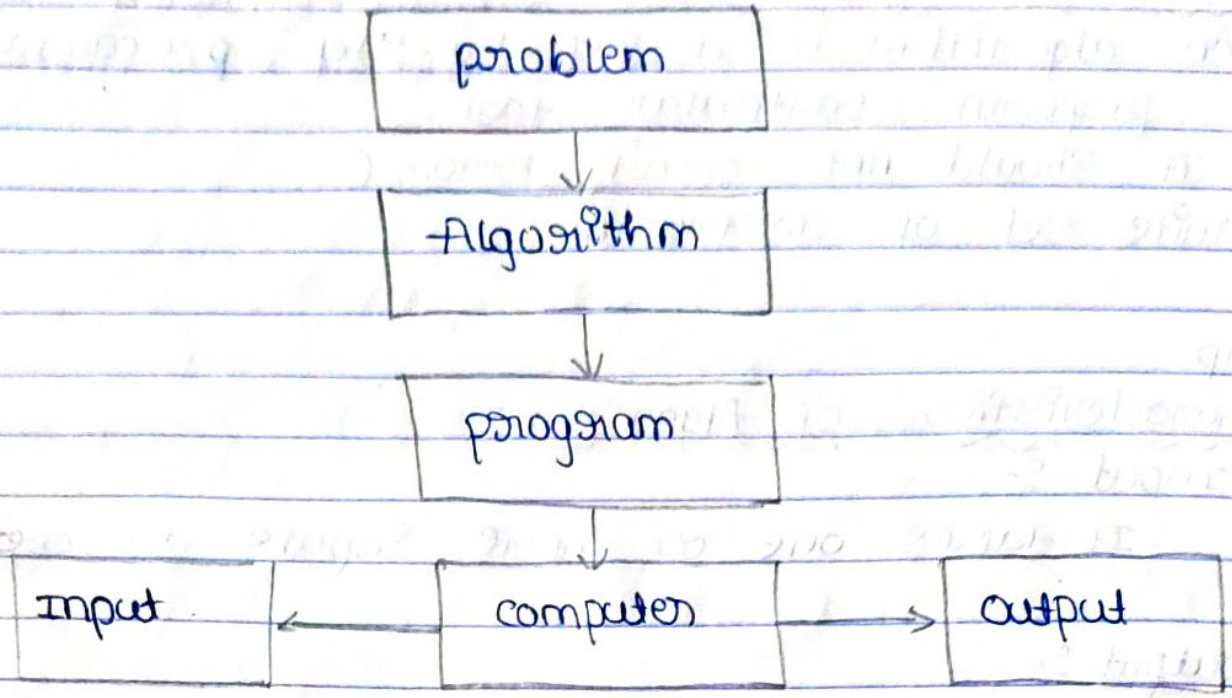
④ Finiteness :-

An Algorithm must terminate after a finite number of steps

⑤ Effectiveness :-

which will take less time and less space

* Diagramatic Representation of Algorithm



① print "Hello word"

code :-

```
# include <stodio.h>
# include < conio.h>
void main ()
{
 printf ("Hello word")
 ~~Scanf~~ getch ();
}
```

Algorithm :-

Step 01 :- Start.

Step 02 :- taking Headerfiles

Step 03 :- ~~Reading~~ Enter Hello word.

Step04 :- Reading Hello word

Step 05 :- Result

Step 06 :- End

② print "This is my first class"

code :-

```
#include< stodio.h>
#include< conio.h>
void main()
{
 printf "This is my first class"
 scanf "
 getch();
 }
```

Algorithm :-

Step 01 :- Start
Step 02 :- taking Headerfiles
Step 03 :- Input "This is my first class
Step 04 :- Result
Step 05 :- End

③ compute c=a-b

code :-

```
#include<stodio.h>
void main ();
{
 int a,b,c;
 printf (" Enter a value for a & b ");
 scanf ("%d ",&a & b);
 sub = a-b;
```

char :- one letter 'with in single court'
Streeng :- group of words 'within double court

Algorithm :-
Step 01 : Start.
Step 02 : taking Header files
Step 03 : Declering the value a, b, c
Step 04 : Enter the value a & b
Step 05 : Reading the values ("%d", &a&b)
Step 06 : Compute the values c=a-b
Step 07 : Result c=a-b
Step 08 : End

properties of an Algorith :-
the algorith satisfies the following conditiga
① input :- each Algorith ~~mu~~ should have 0 or 1 input
② output :- Function which returns 0 or 1 (void, int)
③ Definitness :- each instruction should be clear & un ambiguiess (Complex number)
④ Effectiveness :- Taking only necessary statements, instruction should be simple and should transform the given input in desire ~~output~~ output.
Ex :- Addition of 2 numbers
⑤ finiteness :- An Algorithm must terminate after finite sequense of instruction

Sorting :- Ex :	Enter 5 values
① Bubble Sort	2, 3, 1, 5, 6, 4
② selection sort	1, 2, 3, 4, 5, 6.
③ Insertion sort	
④ Quick sort.	
⑤ Merge sort	

Agorithm
Implimentation

searching
① Linear
② Binary.

take the problem understanding
Implimention
front n.
back n.

collect the data

problem solving Aspets Aspects

① problem defination
② getting started one problem
③ use of specific example
④ similarities among problems
⑤ working backwards from solⁿ the solⁿ

→ ① problem defination :-
In solving any problem it is only possible after we have made the effort to come to terms with are understand the problem at hand.
Ex :- student management system
this is important becouse a clear under-standing of the problem will guide the rest of the process & help you design and effective solⁿ
* what is importance of understanding the problem
* technics for breaking down the complex problems.
* Identifie the input and output

② getting started on problem :-
~~yet~~ they are many wage to solve the problem and also many solⁿ to the problems
The situation doesn't make the job of problems solving easy.
Some times it is deficult to recagnize quikly which paths are likely to be good.

and which paths may be productive.

③ use of specific example:- for example

If you want to find the maximum in a set of number choose particular set of number & workout the machanisum for finding maximum in the set.

Ex :- finding the maximum among two number student or employee.

④ similarity among problems :-

We already seen that one way to make start on program is by considering specific example

* In this respect it is importancy ~~If~~ to see if there any similarity btn current problem & ontherr problem.

that we have seen or saved.

A good habitte they for is to always make an effort to be aware of similarity among problems.

Ex :- ~~Anstince~~ Einstein's descovery of relativity, and newton's theory.

⑤ working backwood from the soln :-

It is useful when the end result is desired.

Working background soln technic where one start from the desired final outcome and ent antipes directs in reverse determine the initial condn or the path needed.

Steps :-

* clarifise series of events ~~of~~ wer leading to a results.

* can reduce the complex city by reversing operations

"We learn most when we have to invent"

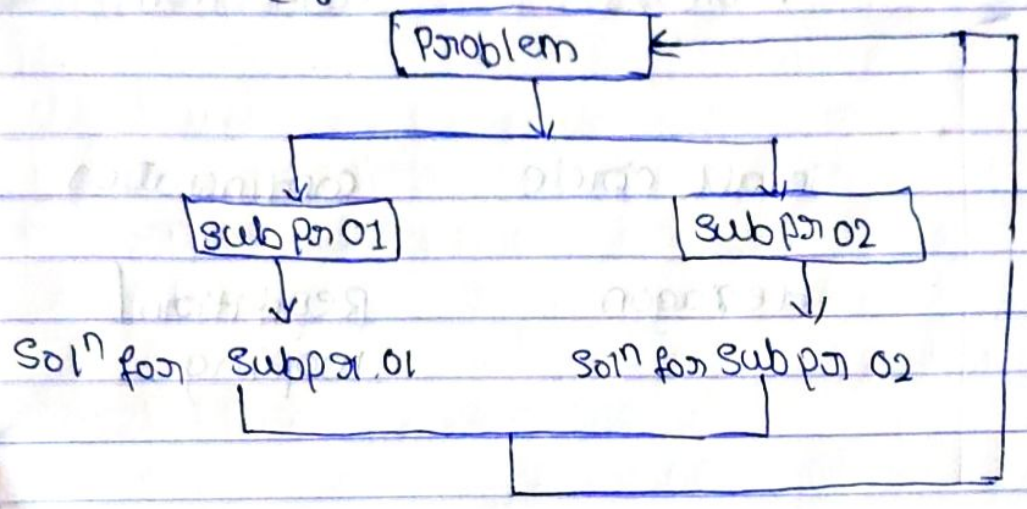
Top-down design (Divide (conqre))
problem solution

The Top down design. It is ~~but~~ works like a Divide (concers stortegy. where thep proble (complex Problem) are divide it into subproblems to find the sol^n

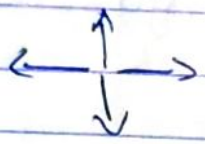
An A technique for Algorithm design they ~~try~~ tries to accumarate human limitat-ion is Known a top down design. (step wise refinement).

top down design helps how to handle complex problems and to find the sol^n.

Ex :- Devide and concer stortergy (quic sort and murge sort)

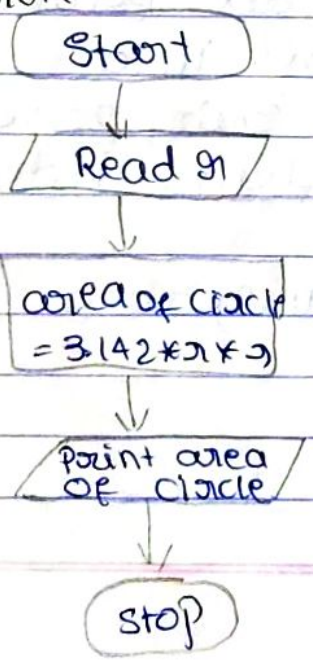


* start with a over all problems
* divide the Problem into sub problems
* forther divide the sub problems.
* solve each problem indurisually combine the sol^n
* refinement & optimizations

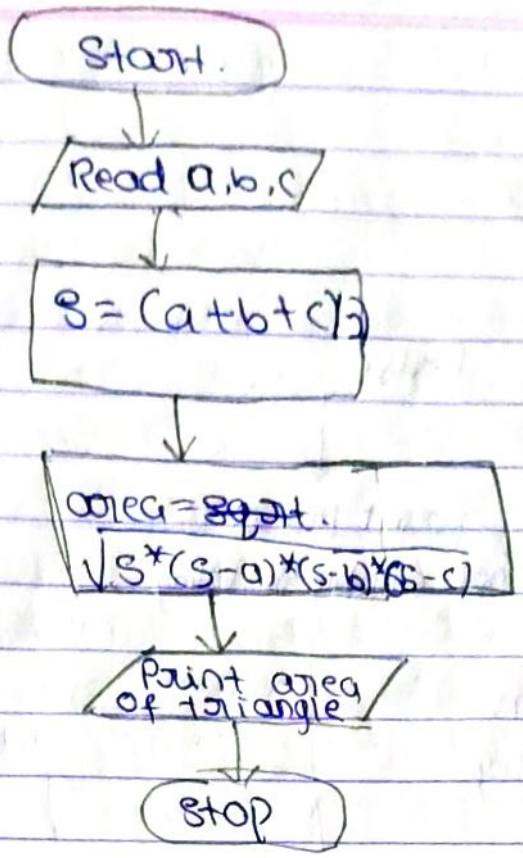
Geometrical Figure	Name	Function
	oval	Start and stop
	Parallelogram	Input and output
	Rectangle	processing
(conditional statement)	Diamond	Decision making
	Arrows	connections
	Small circle	continuation
	Hexagon	Repitition/ looping.

Example :-

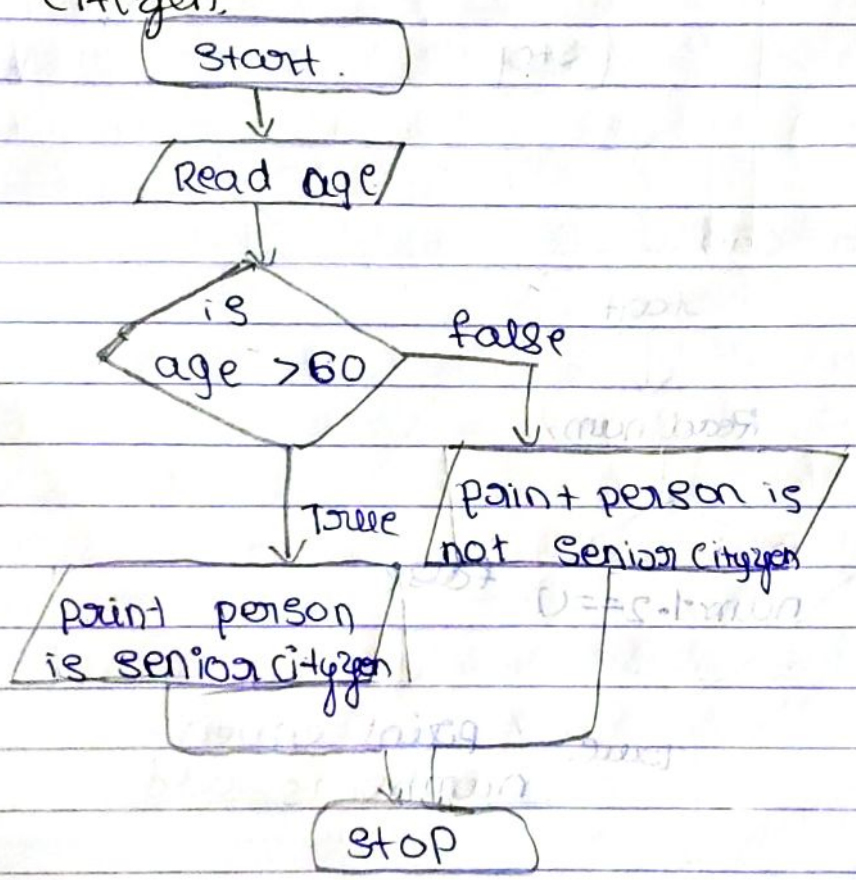
Area of circle

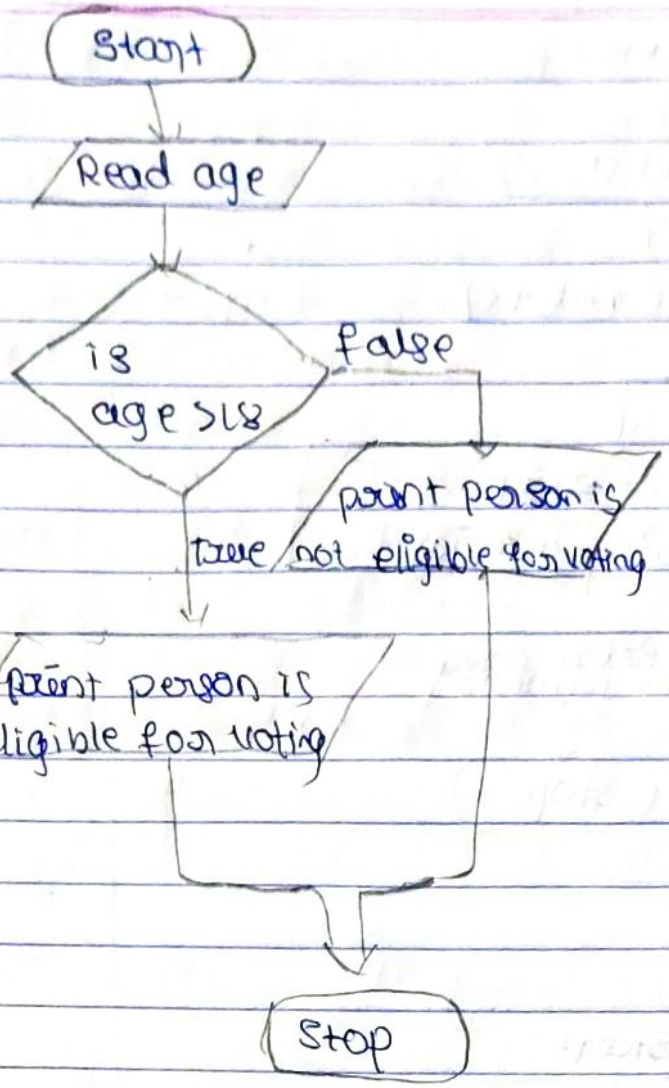


* Area of triangle.

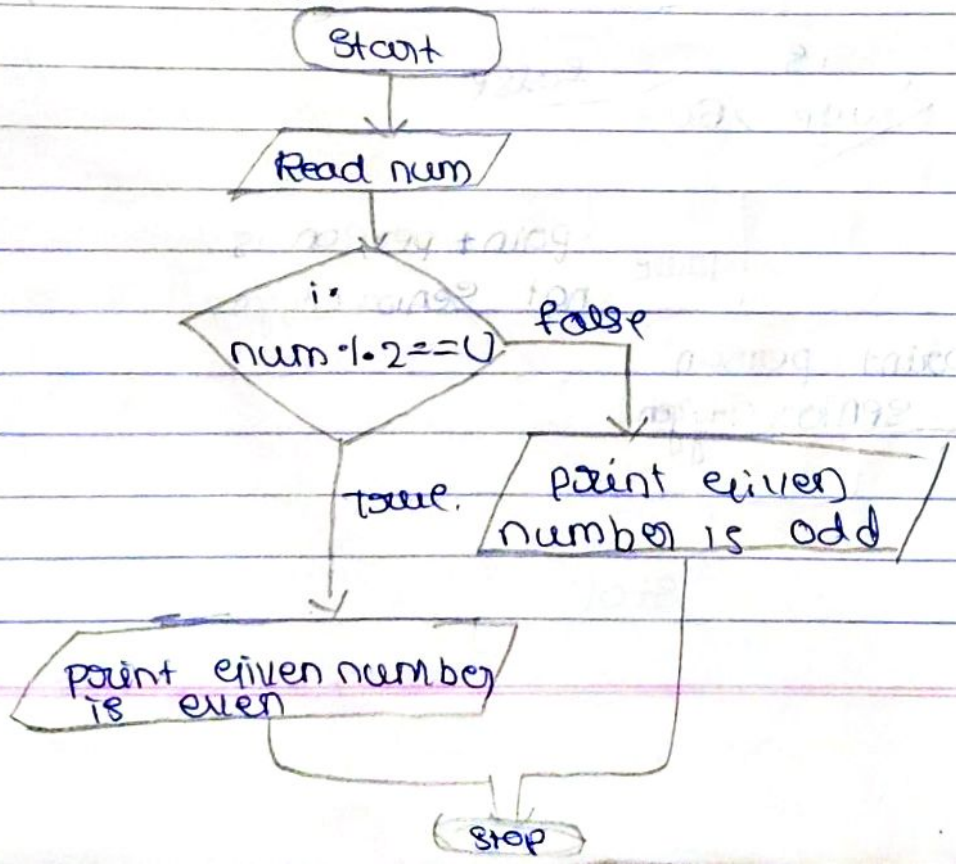


* Sineor citizen.



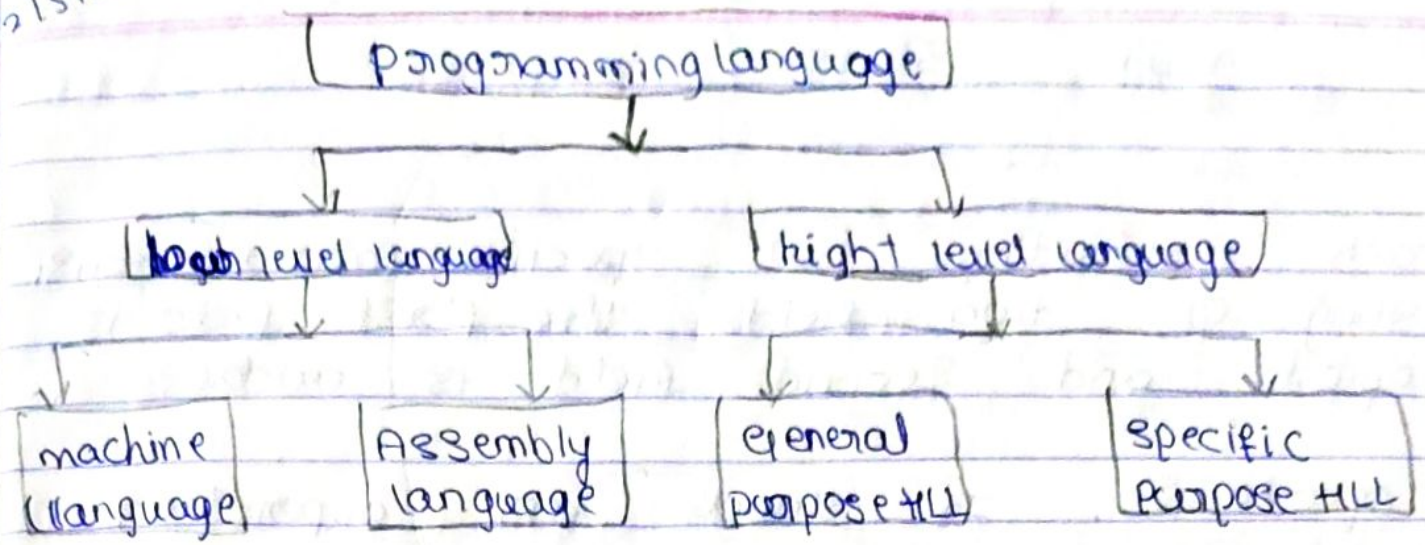

Start
Read age
is age >18
false
print person is not eligible for voting
true
print person is eligible for voting
Stop

even or odd


Start
Read num
is num % 2 == 0
false
print given number is odd
true.
print given number is even
Stop

computer languages :

** -
21/5/10



compute does not understand english like language. It understands and process any thing that is in the form of 0s and 1s the 0's & 1's are alphabets in computer readable languages.

computer languages are cassified into low level programming languages & high level

The low level languages are those languages which are easily understode by the machines.
* The highe level languages are those which are nearer to the natural languages such as english.

machine language :-
As the name it self implise programming is done of machine language the vocabullary of computer consist of only two words 0 & 1 that is the digits of binary number system. A zero indicates low voltage level & one indicates high volt
the machine level language is consider as the first generation language

opcode	Address
4 bit	8 bits

each instruction has specific formate consi-sting of two fields the first field is opcode and second field is address

* opcode :- It stands for opperation code It indicate what opperation is to be perform the opperation could be Addition, substraction, multiplication, store, halt.

* Address :- Memory location were in the data is referenced.

the length of each instruction is 12 bits → 4 bits for opcode, 8 bits for address

Ex :- | 1010 | 10100101 |

Advantages
* can be directly typed and exicuted.
* NO translator program is required.

Dis- Advantages
* deficult to remember machine Instruction
* deficult to understand, modified, De-bug errors.
* deficult to remember address of data & Instructions.
* lack of voltability → The program that runs on one machine does not run on other

Implimantation of Algorithms :-

Steps involved in implimantation of Alogorithm

① use of procedure to Emphasize modularity (Define the problem)
② Choice of variable names
③ documantation of programs
④ De-bugging the programs
⑤ Program testing

① use of procedure to Emphasize modularity (Define the problem)

It concentrate to words top-down design define the propblem to be solved input handel, concetrate, expected output with out any deficulty

Ex :- Addition of two number

② Choice of variable names.

variable name should be suitable each variable should only have one role In a given program

③ Documantation of programs

~~④ De-bugging the programs~~

Starts with begin should gives the bracket {}, they should be consise but accuratly specifise what is required.

Ex :- C-program

④ De-bugging the program

In Emplimanting the Algorithm many test should be ~~down~~ done to chek whether it is working properly or not

⑤ Program testing

Need to check all the condition will it work properly.

* program should even respond for the undue - op. condition or question which is not developed for it.

program	space.
time	space
stack	space

counting semestion of ~~a~~ set of a number:-

```
#include<stdio.h>
int main()
{
    int ~~count~~ num, i;
    printf(" From 1 to 4 numbers");
    for(i=1; i<=4; i++)
    {
      printf("Enter 1 to 4 number", i);
      sum = sum+i
    }
}
}
```

the semestion of set 'S' containing number $x_1, x_2, x_3 --- x_n$ represented as ~~sen~~ $\sum_{i=n}^{n} x_i$

Ex: $S = \{1, 2, 3, 4\}$

$= 1+2+3+4 = 10$

$= 10$

Notes

PAGE NO.:
DATE:

1. Implimentation of Algorithms

→ It follows top to bottom pattern
→ Implemented in this way are usually much easier to understant and debug

1. use of procedures to emphasize modula-rity?
→ concentrating towards top to down design and implementing functions (procedures). by using functions in out code it can accomplished a prog easily without any difficulty.

2. choice of variable names?
→ variable name should be suitable
* each variable should only have one role in a given program

3. Documentation of programs?
→ Starts with begin, should use brac-ks { }, they should be concise but accurately specity what is valuge

Sulekha

4. Debugging the programs?

→ In implementing algorithm, many tests should be done to check whether it is working properly or not

→ Some times there should be logical errors that donot show up in the compication phase.

→ for this good idea, can used boolean variable that is true or false

→ should work with handle before even attempting to execute it

→ A good rule to follow when debugging is not to assume anyth

5 program testing?

→ need to check all condetions will work properly.

→ program should even respond fo The question which it is not developed for it

→ should design algorithms to be very general so that they will handle a whole class of program rather than just one specific cas

Ex:- while $i<100$ ✗

while $i<n$ ✓

Program verification :-

PAGE NO.:
DATE: / /

→ When it comes to development of software or any applications plays an important role.

→ Especially at the field of military, space and madical applications it is matter of life and death.

→ program verification refers to the application of mathematical proof techniques tof establish that the results obtained by the excution of a program with arbitrary inputs are in accord with formally defined output specifications

Sulekha

factorial computation :-

fact (10)

```
int fact (int 5)
{
  if (n==0)
   (5==0)

  Else
     n* fact (n-1)
     5* fact(5-1)
     5* fact (4)
}
```

best case Efficiency :-
defends on Inputs. It will take will exicute fastly

worst case
defends on Iputs. It will give more Inputs. It will take more time

average case
It will give more than more inputs
It will take much time

Sorting :- The process of arranging a collection of data elements into a specific order, such as ascending or descending, based on a particular key or attribute

① Sum of two numbers

```
#include <stdio.h>
int main()
{
    float a, b, c;
    printf("Enter the value of a");
    scanf("%f", &a);
    printf("Enter the value of b");
    scanf("%f", &b);
    c=a+b;
    printf("c=%f", &c);
}
```

② multiplication of two variables

```
int c)
#include<stdio.h>
int main()
{
    int a, b, mul;
    printf("Enter the value of a");
    scanf("%d", &a);
    printf("Enter the value of b");
    scanf("%d", &b);
    mul=a*b
    printf("mul=%d", &mul)
```

Step 1 :- start.

Step 2 :- taking ~~three~~ two numbers

① Why we need Algorithm.?

① clarity & simplycity

② efficience

③ Reusability

④

② Advantages

* simplycity
* easy to understand
* Error reduction
* Documentation

Dis - Advantages

* Time - consuming to design
* not always practicle
* No standard format.

*c program to desplay Students Information

```
#include<stdio.h>
int main()
{
  int m1, m2, m3;
  printf("Enter the value of m1");
  scanf("%d", &m1);
  printf("Enter the value of m2");
  scanf("%d", &m2);
  printf("Enter the value of m3");
  scanf("%d", &m3);
  printf("%d \n", m1);
  printf("%d \n", m2);
  printf("%d \n", m3);
  getch()
}
```

H.W. - Algorith

Algorithm

Step 01 :- Start

Step 02 :- taking Header files

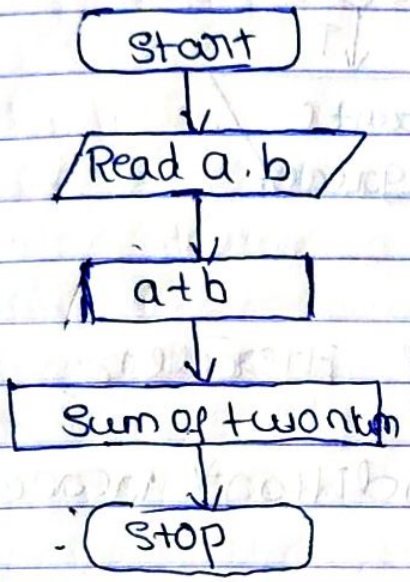
Step 03 :- Declare the value m_1, m_2, m_3

Step 04 :- Read the value ("%d", &m_1, &m_2, &m_3)

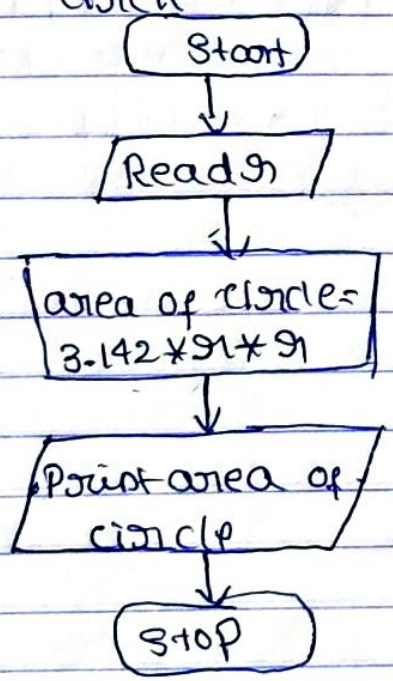
Step 05 :- Result

Step 06 :- End.

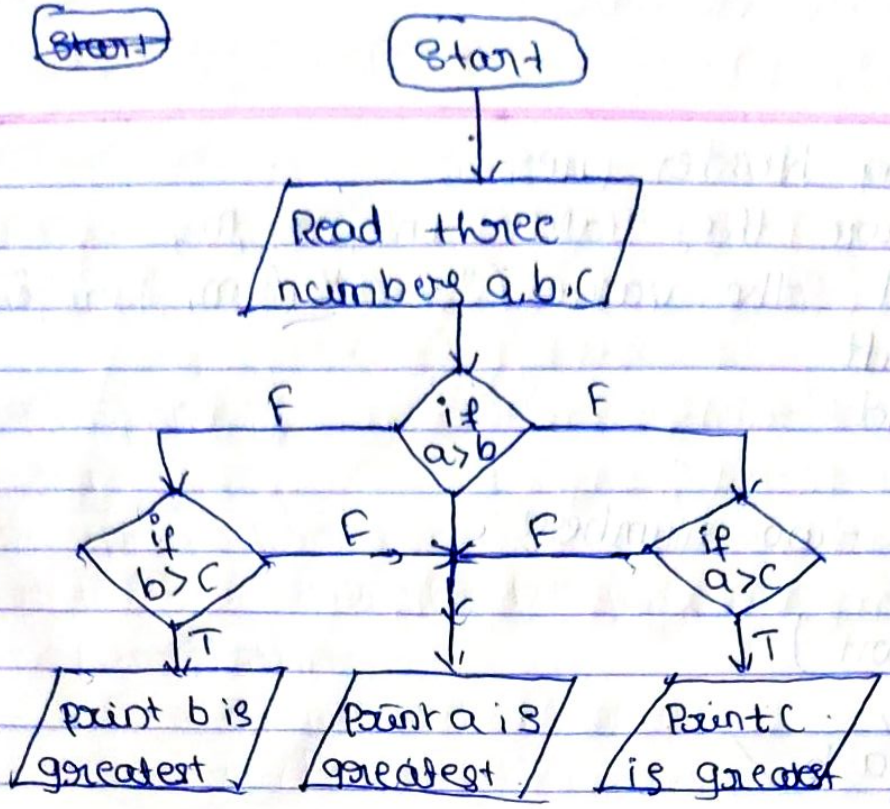
* Addition of two numbers



* Area of Circle



* to find greatest of ~~numb~~ three numbers



for - to find number of iterations
3 - parameters
* Initilization * condition * increament/decrement

① Data type in C

* Basic data types
① int - %d
② float - %f
③ char - %c
④ bool
⑤ void

* Derived Data types
① array
② pointer
③ function

* user defined Data types.
① union
② Structure
④ enum

② opperators
① Arithmetic operators (+, -, *, /, %)
② comparison operators (==, !=, >, <, >=, <=)
③ logical operators (&& (and), || (or), ! (not))
④ Assignment operators (=, +=, -=, *=, /=)
⑤ Bitwise operators (&, |, ^, ~, <<, >>)

③ conditional opes Statement
→ A conditional Statement is a way to execute certain code only if a condition is true

* Simple if
* if-else
* if-elif-else (in python)
* nested if
* switch-case (in languages like C, Java)

Looping Statements

* A looping statement is used to repeat a block of code multiple times, until a condition is false (or true, depending on the type of loop).

Types of Loops

1. for loop
2. while loop
3. do-wile loop

* Algorithm for Exchanging values with using temp variable

Step 1:- Start.
2:- taking Header files
3:- declare the variables a, b, c
4:- Enter the two numbers
5:- Reading the values a, b
6:- before the swaping a = 1.d ln
b = 1.d ln , a & b
7:- compute logic temp = a
a = b
b = temp
8:- after swaping a = 1.d ln, b = 1.d ln, a, b.
9:- compute the result
10:- End

$\frac{10}{2} = 10$

$\frac{4 * 5}{2}$

$\frac{4 \times (4+1)}{2}$

$4 * (4+1) / 2$
$1 * (2) / 2$
$2 / 2 = 1$

$2 * (2+1) / 2$
$2 * (3) / 2$

0+1
1
1+1
2
2+1
3
3+1
4

for(i=1 ; i<=4; i++)

i=1 ; i<=1 ; i++

i=0 sum = 0+1 i=1 = sum=0+1

= 1

i=2 sum = 1+2 sum=

= 3

i=3 sum = 3+3

= 6

i=4 sum = 3+4

= 7

(2) sum of natural numbers (10)

(2) counting the semetion of a set a number.

① write a C program to find sum of natural number

```
#include<stdio.h>
int main()
{
 int n=4; sum=0;
 for(int i=1; i<=n; i++)
 {
  sum=i; or sum=sum+i or count=count+i;
 }
 printf("The sum of the first %d natural
        numbers is: %d\n", n, sum);
 return0;
}
```

short formula method (n*(n+2)/2)

* Counting summation of set of numbers

4 variables
a[i] = array
n = number
sum = store the va
i = increment decrement

```
#include <stdio.h>
{
  int a[10], i, sum=0, ;
  printf("Enter the n numbers");
  scanf("%d", &n);
  printf("Enter the set of values");
  scanf("%d", &a[i]);
  for(i=0; i<=n; i++)
  {
      sum=sum+i;
  }
}
```

~~Step01 :- Start~~
Step02 :- declaring the variables a[10], i, sum=0, n
Step03 :- Enter the n value
Step04 :- Reading the values n
Step05 :- write the logic for(i=0; i<=n; i++)
Step06 :- ~~count the~~ sum the values
sum = sum+i

~~Step 07~~ :- Algorithm :-
Step 01 :- Start
Step 02 :- declaring the variables a[10], i, sum=0, n
Step 03 :- Enter the n value.
Step 04 :- Reading the n values
Step 05 :- Enter the set number
Step 06 :- Read the values "%d", &a[i]
Step 07 :- ~~Count~~ ~~logi~~ write the logic for (i=0; i<=n; i++)
Step 08 :- Sum the values
sum = sum+i.
Step 09 :- compute the value
Step 10 :- End.

Swapping upon two number without using theird variable

a= a+b;
b= a-b;
a= a-b;

Ex = a=2 a= 2+3=5
b=3 b= 5-3=2
a= 5-2=3

after swapping a=3
b=2

① Swapping two number without using third variable.

Program.

```
#include<stdio.h>
void main()
{
  int a,b;
  print("Enter the two values");
  scanf("%d", &a, &b);
  printf(" Before swapping \n a=%d\n, b=%d\n",
          a,b);
  a=a+b;
  b=a-b;
  a=a-b;
  printf(" After swapping\n a=%d\n, b=%d\n,
          a,b);
}
```

Algorithm :-

Step 01 :- Start
Step 02 :- declaring the two variables a,b
Step 03 :- Enter the two values a & b.
Step 04 :- Reading the a & b values
Step 05 :- Before swapping ~~a=%d~~ \n, b=%d\n,
a,b);
Step 06 :- compute a=a+b
b =a-b
a =a-b

Step 07 :- After swapping ~~b=a~~ a=1.d\n, b=1.d\n, a, b

Step 08 :- compute the Result

Step 09 :- End

a = 5 a = 5+10 = 15

b = 10 b = 15-10 = 5

a = 15 - 5 = 10

a = 5

a = 10

15 & 35

a = 15 a = a+b = 15+35 = 50

b = 35 b = a-b = 50-35 = 15

b = 15 a = a-b = 50-15 = 35

a = 35.

<u>function</u> :- if it is followed by the bracket ()

() ; → Declaration

<u>Iteration</u> :- <u>(loop's)</u>

for(i=0 ; i<n ; i++)

n=2

for(i=0 ; 0<2 ; i++)

T

for(i=1 ; 1<2 ; i++)

T.

for (i=2 ; 2<2 ; i++)

f

for (i=0; i<n; i++)

~~for if (4==0)~~ I $i=0$ $n=2$

$0<2$

T

$i=i+1$

$=0+1$

$=1$

II $i=1$ $1<2$

T

$i=i+1$

$=1+1$

$=2$

III $i=2$ $2<2$ f.

for (i=0; i<n; i++)

$i=0$ $n=4$

$0<4$ T $i=i+1$

$=0+1$

$=01$

$i=1$ $n=4$

$1<4$ T $i=i+1$

$=1+1$

$=2$

$i=2$ $n=4$

$2<4$ T $i=i+1$

$=2+1$

$=3$

$i=3$ $n=4$

$3<4$ T $i=i+1$

$=3+1$

$=4$

$i=4$ $n=4$

$4<4$ f. for (i=4; 4<4; i++) false

fact = 1 n = 2

if (n == 0)

2 == 0

return 1

Else

for (i = 1; i <= 2; i++)

i = 1 n = 2

1 <= 2 = T.

{

fact = fact * i;

= 1 * 1

= 1

fact = 2 n = 2

if (n == 0)

2 == 0

return 1

Else

for (i = 2; i <= 2; i++)

i = 2 n = 2

2 <= 2 = T

{

fact = fact * i;

= 1 * 2

= 2

* n = 5 i = 1 fo

fact = 1

if (n == 0)

5 == 0 ✗

return 1

Else

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

i = 1 n = 5

1 <= 5 T

{

fact = fact * i;

= 1 * 1

= 1

for (i = 3; i <= 5; i++)

i = 3, n = 5

3 <= 5 T

{

fact = fact * i;

= 2 * 3

= 6

if for (i = 2; i <= 5; i++)

i = 2 n = 5

2 <= 5 T

{

fact = fact * i;

= 1 * 2

= 2

for (i = 4; i <= 5; i++)

i = 4, n = 5

4 <= 5 T

{

fact = fact * i;

= 6 * 4

= 24

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

i = 5, n = 5

5 <= 5 T

{ fact = fact * i; 24 * 5

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17 19
20 21 22 23 24 26 27 28 29 30 31 32 33 34
37 38 39 41 42 43 45 46 47 50 51 52 53 55
56 58 62 63 64 65 59 54

* Factorial with using iteration

```
int fact (int n)
{
    if (n==0)
  else
      n* fact(n-1)
```

n=5

```
int fact (int5)
{
  if (n==0)
     (5==0) ✗
  else
   n* fact (n-1)
    5* fact (5-1)
    5* fact (4)
```

```
int fact (int4)
{
  if (n==0)
  if (4==0) ✗
  else
   n* fact(n-1)
    4* fact(4-1)
    4* fact(3)
```

```
int fact (int3)
{
  if (n==0)
     (3==0) ✗
  else
   n* fact(n-1)
    3* fact(3-1)
    3* fact(2)
```

```
int fact (int 2)
{
  if (n==0)
    (2==0) ✗
  else
   n* fact(n-1)
    2* fact(2-1)
    2* fact(1)
```

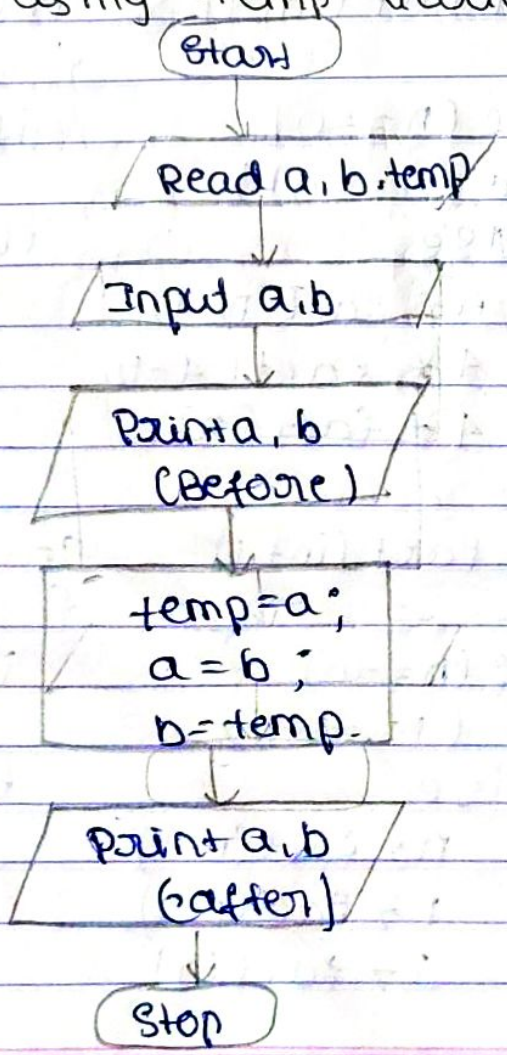
```
int fact (int 1)
{
  if (n==0)
    (1==0) ✗
  else
   n* fact(n-1)
    1* fact(1-1)
    1* fact(0)
```

```
int fact (int 0)
{
  if (n==0)
    (0==0) ✓
  ~~else~~ 1
  ~~0*~~
```

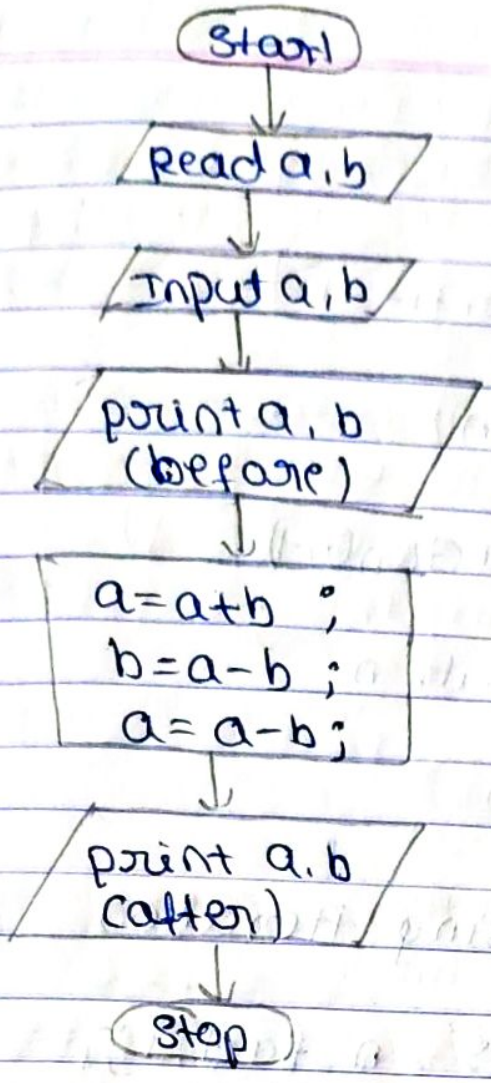
* factorial with out using Iteration.

```
#include <stdio.h>
int main()
{
    int n;
    printf("Enter the value of n");
    scanf("%d", &n);
    int fact (int n)
    {
        if (n==0)
    else
        n* fact(n-1)
    }
}
```

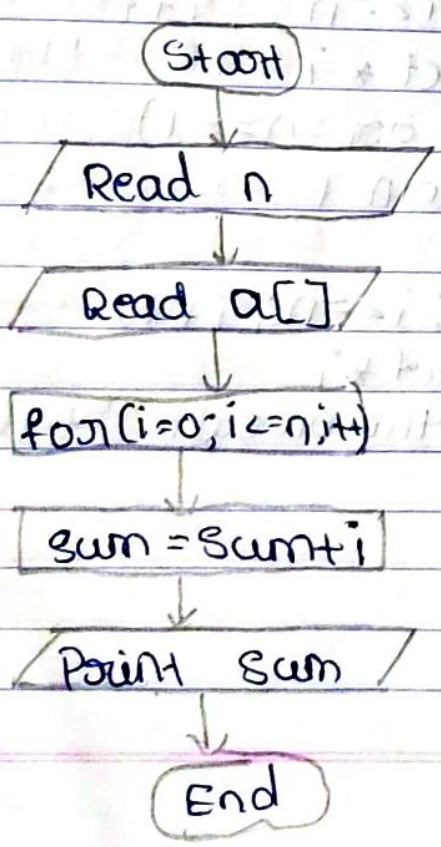
* with using temp variable



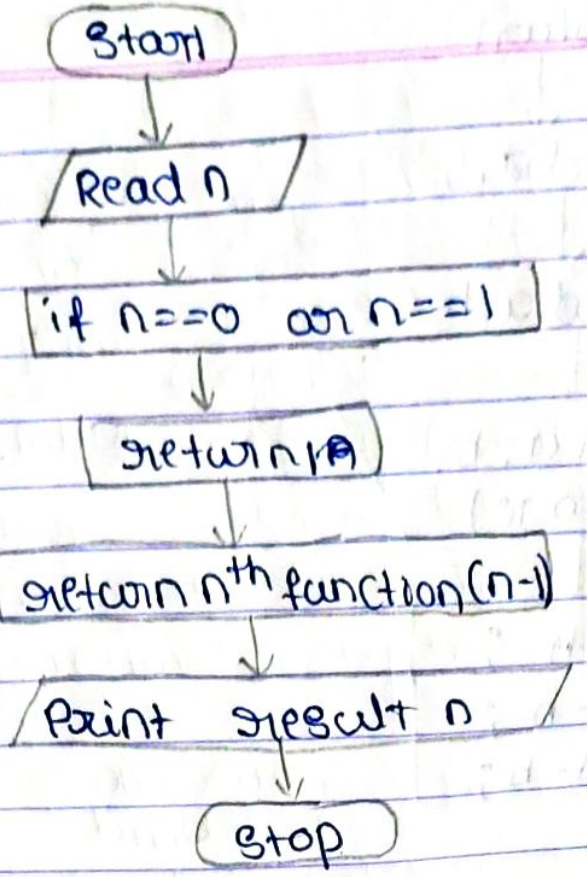
* with out using temp variable



* Summation of set



* factorial



* factoria without using iteration

Step 01 :- Start
Step 02 :- Declare variables n, fact, i;
Step 03 :- Input the values i
Step 04 :- Read the condition
Step 05 :- Checking condition

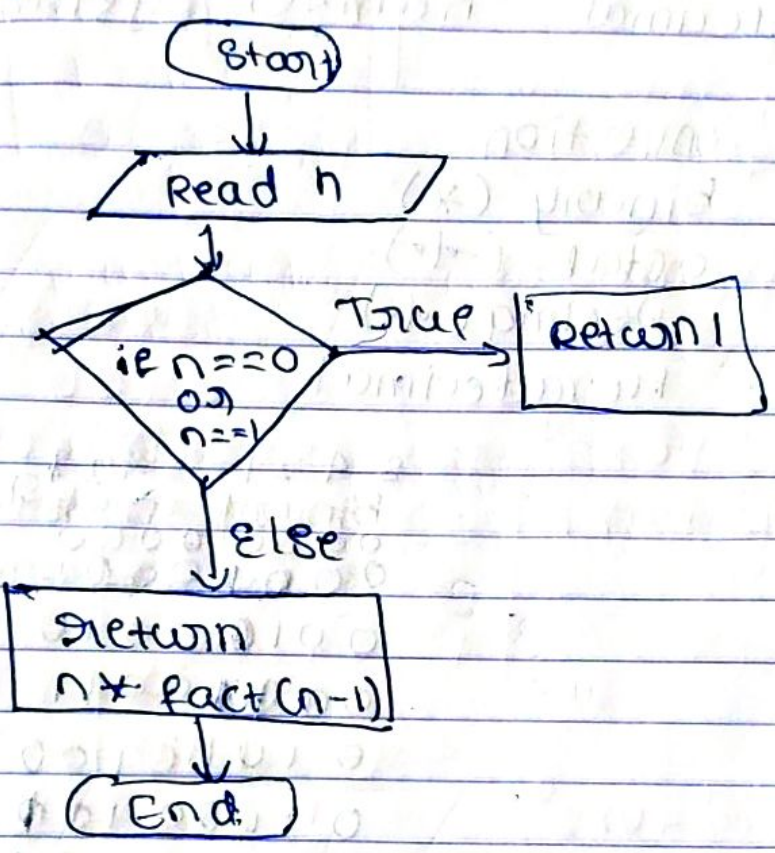
```
for (i=0 ; i<=n ; i++
    fact = fact * i
    if (n==0 or n==1)
        return 1
    Else
    for(i=0 ; i<=n ; i++)
    fact = fact * i
```

Step 06 :- compute the result "i.d=i"
Step 07 :- End

1 - 10 binary

Iteration	Recursion
~~* for a n~~	
* n number of calculation	* It checks again and again It self
* loop condition	* Base case
* ~~Higher~~ lower overhead (more memory Efficient)	* Higher overhead (stack usage function calls

* factorial.



* Ascii value

The ASCII value of 'A' is 65 and it Increases sequentially to 90 for 'Z'.

uppercase A-Z.

A=65, B=66, C=67, D=68, E=69, F=70, G=71 H=72, I=73, J=74, K=75, L=76, M=77, N=78, O=79, P=80, Q=81, R=82, S=83, T=84, U=85, V=86, W=87, X=88, Y=89, and Z=90

Lowercase a-z

a=97, b=98, c=99, d=100, e=101, f=102, g=103.

h=104, i=105, j=106, k=107, l=108, m=109, n=110

o=111 p=112 q=113 r=114 s=115 t=116

u=117 v=118 w=119 x=120 y=121 z=122

Number system Conversion

* decimal number system (0-10
* octal number system (0-7)
* bainary number system (0 1)
* Hexa decimal number system

Base ~~and~~ conversion

2 (0.1)	binary (*)
8	octal ~~(10)~~
10	Decima (10)
16	Hexadecimal

Decimal numbr	8 4 2 1 ~~64 27~~ ~~54~~ Binary numbr.
0	0000 ~~0000~~
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1010
10	1010
11	1011

2^{100}

Decimal	Binary (8 4 2 1)
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110

* Binary to Decimal

0101

~~2 X 0 + 2 X 1 + 2~~

$0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$0 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1$

$0 + 4 + 0 + 1$

$4 + 1 = 5$

$= (5)_{10}$

1010

$= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

$= 1 \times 8 + 0 \times 4 + 1 \times 2 + 0$

$= 8 + 0 + 2 + 0$

$= (10)_{10}$

1011

~~2 X~~ $1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

$1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1$

$8 + 0 + 2 + 1$

$= (11)_{10}$

8 4 2 1

1 1 0 0

$1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

$1 \times 8 + 1 \times 4 + 1 \times 2 + 0 \times 1$

$8 + 4 + 2 + 0$

$= \cancel{14}\ (2)\ (14)_{10}$

* base

* L-R

* 8421

* Decimal$_{(10)}$ to Binary$_{(2)}$

2	10	
2	5	- 0
2	4	- 1
2	2	- 1
	1	- 0

2	10	
2	5	→ 0
2	4	→ 1
2	2	→ 0
	1	→ 0

2	6	
2	3	0
	1	1

1011

2	11	
2	5	1
2	2	1
	1	0

2	6	
2	3	0
	1	0

2	6	
2	3	0
	1	1

2	3	
	1	

2	5	
2	2	1
	1	1

* Binary$_{(2)}$ to Decimal$_{(10)}$

① 010
$0\times2^3+0\times2^2+1\times2^1+0\times2^0$
$0\times8+0\times4+1\times2+0\times1$
$= 0+0+2+1$
$= 2_{(2)}$

② 0100
$0\times2^3+1\times2^2+0\times2^1+0\times2^0$
$0\times8+1\times4+0\times2+0\times1$
$0+4+0+0$
$= 4_{(2)}$

③ 1100
$1\times2^3+1\times2^2+0\times2^1+0\times2^0$
$1\times8+1\times4+0\times2+0\times1$
$8+4+0+0$
$= 12_{(2)}$

④ 1101
$1\times2^3+1\times2^2+0\times2^1+1\times2^0$
$1\times8+1\times4+0\times2+1\times1$
$8+4+0+1$
$= 13_{(2)}$

⑤ 1110
$1\times2^3+1\times2^2+1\times2^1+0\times2^0$
$1\times8+1\times4+1\times2+0\times1$
$8+4+2+0$
$= 14_{(2)}$

* Decimal$_{(10)}$ to Binary$_{(2)}$

① 1001

2	9	
2	4	1
2	2	0
	1	0

$= 1001_{(2)}$

② $10_{(10)}$

2	10	
2	5	0
2	2	1
	1	0

$= 1010_{(2)}$

③ $11_{(10)}$

2	11	
2	5	1
2	2	1
	1	0

$= 1011_{(2)}$

④ $12_{(10)}$

2	12	
2	6	0
2	3	0
	1	1

$= 1100_{(2)}$

⑤ $13_{(10)}$

2	13	
2	6	1
2	3	0
	1	1

$= 1101_{(2)}$

$5_{(10)}$

2	5	
2	2	1
	1	0

$= 0101_{(2)}$

2	7	
2	3	1
	1	1

$0111_{(2)}$

2	6	
2	3	0
	1	1

$0110_{(2)}$

2	973	
2	431	111
2	21	01
	1	0

111010

$1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1$

$8 + 4 + 1$

$= 13$

Perform base Conversion

1) $(1101)_2$ 2) $(973)_{10}$

program set of Instruction to exicuting perticular

sorting :-
Sorting is the nbr of

Decimal to octal.

$(973)_{10} = (1715)_8$

8	973	
	121	5
	~~15~~ 15	1
	1	7

8	973	
8	121	5
8	15	1
	1	7

$(654)_{10} = (1216)_8$

8	654	
8	~~64~~ 81	6
8	10	1
8	8	2
	1	

$(104)_{10} = (\quad)$

8	104	
8	18	0
	2	2

octal to decimal

$(653)_8 \longrightarrow (?)_{10}$

$= 6\times8^2 + 5\times8^1 + 3\times8^0$

$6\times64 + 40 + 3$

$384 + 40 + 3$

$= (427)_{10}$

$(7014)_8 \longrightarrow (?)_{10}$

$= 7\times8^3 + 0\times8^2 + 1\times8^1 + 4\times8^0$

$= 7\times512 + 0\times64 + 1\times8 + 4$

$= 3584 + 0 + 8 + 4$

$= 3596$

$(627)_8$

$6\times8^2+2\times8^1+7\times8^0$

$6\times64+2\times8+7\times1$

$384+16+7$

$= \cancel{397}\ 407$

$(370)_{10}$

$3\times8^2+7\times8^1+0\times8^0$

$3\times64+7\times8+0\times1$

$= 192+56+0$

$= (248)_8$

384
16
1 7
16 7

* Decimal - octal

(1) $(653)_{10} \longrightarrow (1215)_8$

8	653	
8	81	5
8	10	1
	1	2

(2) $(627)_{10} \longrightarrow (1163)_8$

6

8	627	
8	78	3
8	9	6
	1	1

B → D → 0110 → *2
D → B → 1 2
0 → D → *8
D → 0 → 1 8
B → 0 → *2

* Binary - octal

$(0110)_2 \longrightarrow (6)_8$

$0\times2^3+1\times2^2+1\times2^1+0\times2^0$

$0+4+2+0$

$= 6$

$(01101)_2$

$0\times2^4+1\times2^3+1\times2^2+0\times2^1+1\times2^0$

$0\times16+1\times8+1\times4+0\times2+1\times1$

$= 0+8+4+0+1$

$= (13)_8$

$(01100)_2 \rightarrow (14)_8$

$0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

$0 \times 16 + 1 \times 8 + 1 \times 4 + 1 \times 2 + 0 \times 1$

$0 + 8 + 4 + 2 + 0$

$= (14)_8$

* octal to Binary

$(41)_8 \rightarrow (100001)_2$

$4 \times 8^1 + 1 \times 8^0$

$= 32 + 1$

$= 33$

2	33	
2	16	1
2	8	0
2	4	0
2	2	0
	1	0

$\frac{3 \times 12}{29}$

$\frac{2 \times 14}{28}$

$\frac{2 \times 15}{30}$

128
56
28
192

$\frac{5 \times 8}{512}$

3584
8
19
3596

Ex. :- * B to D

$(1010)_2 \rightarrow (10)_{10}$

$= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

$= 1 \times 8 + 0 \times 4 + 1 \times 2 + 0 \times 1$

$= 8 + 0 + 2 + 0$

$= (10)_{10}$

8×10
80

* D $\rightarrow$ B

$(53)_{10} \rightarrow (110101)_2$

$\frac{2 \times 15}{30}$

$tlw = 12 \times 5$
$= 60$

2	53	
2	26	1
2	13	0
2	6	1
2	3	0
	1	1

* $O \rightarrow D$

* $(278)_8 \longrightarrow (192)_{10}$

$2 \times 8^2 + 7 \times 8^1 + 8 \times 8^0$

$2 \times 64 + 7 \times 8 + 8 \times 1$

$= 128 + 56 + 8$

$= 192$

$(7014)_8 \longrightarrow (3596)_{10}$

$7 \times 8^3 + 0 \times 8^2 + 1 \times 8^1 + 4 \times 8^0$

$7 \times 512 + 0 \times 64 + 1 \times 8 + 4 \times 1$

$3584 + 0 + 8 + 4$

$= (3596)_{10}$

* $D \rightarrow O$

$(725)_{10} \longrightarrow (1337)_8$

8	725	
8	91	7
8	11	3
	1	3

* $O \rightarrow B$

$(76)_8 \longrightarrow (111110)_2$

$7 \times 8^1 + 6 \times 8^0$

$56 + 6$

$= 62$

2	62	
2	31	0
2	15	1
2	7	1
2	3	1
	1	1

* $B \rightarrow O$

$(1110)_2 \longrightarrow (14)_8$

$1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

$= 1 \times 8 + 1 \times 4 + 1 \times 2 + 0 \times 1$

$= 8 + 4 + 2 + 0$

$= 14$

* $O \rightarrow D$

$(37)_8 \longrightarrow (31)_{10}$

$3 \times 8^1 + 7 \times 8^0$

$3 \times 8 + 7 \times 1$

$24 + 7$

$= 31_{(10)}$

* Binary to Decimal.

① $(1011)_2 \rightarrow (11)_{10}$

$= 1\times2^3+0\times2^2+1\times2^1+1\times2^0$

$= 1\times8+0\times4+1\times2+1\times1$

$= 8+0+2+1$

$= (11)_{10}$

8 4 2 1
1 0 1 1
(11)

② $(1100)_2 \rightarrow (12)_{10}$

$= 1\times2^3+1\times2^2+0\times2^1+0\times2^0$

$= 1\times8+1\times4+0\times2+0\times1$

$= 8+4+0+0$

$= (12)_{10}$

8 4 2 1
1 1 0 0
(12)

③ $(1110)_2 \rightarrow (14)_{10}$

$= 1\times2^3+1\times2^2+1\times2^1+0\times2^0$

$= 1\times8+1\times4+1\times2+0\times1$

$= 8+4+2+0$

$= (14)_{10}$

8 4 2 1
1 1 1 0
(14)

④ $(0111)_2 \rightarrow (7)_{10}$

$= 0\times2^3+1\times2^2+1\times2^1+1\times2^0$

$= 0\times8+1\times4+1\times2+1\times1$

$= 0+4+2+1$

$= (7)_{10}$

8 4 2 1
0 1 1 1
(7)

⑤ $(1010)_2 \rightarrow (10)_{10}$

$= 1\times2^3+0\times2^2+1\times2^1+0\times2^0$

$= 1\times8+0\times4+1\times2+0\times1$

$= 8+0+2+0$

$= (10)_{10}$

⑥ $(1101)_2 \rightarrow (13)_{10}$

$= 1\times2^3+1\times2^2+0\times2^1+1\times2^0$

$= 1\times8+1\times4+0\times2+1\times1$

$= 8+4+0+1$

$= (13)_{10}$

⑦ $(1001)_2 \rightarrow (9)_{10}$

$= 1\times2^3+0\times2^2+0\times2^1+1\times2^0$

$= 1\times8+0\times4+0\times2+1\times1$

$= 8+0+0+1$

$= (9)_{10}$

⑧ $(0100)_2 \rightarrow (4)_{10}$

$= 0\times2^3+1\times2^2+0\times2^1+0\times2^0$

$= 0\times8+1\times4+0\times2+0\times1$

$= 4+0+0$

$= (4)_{10}$

⑨ $(1111)_2 \rightarrow (15)_{10}$

$= 1\times2^3+1\times2^2+1\times2^1+1\times2^0$

$= 1\times8+1\times4+1\times2+1\times1$

$= 8+4+2+1$

$= (15)_{10}$

⑩ $(0101)_2 \rightarrow (5)_{10}$

$= 0\times2^3+1\times2^2+0\times2^1+1\times2^0$

$= 0\times8+1\times4+0\times2+1\times1$

$= 0+4+0+1$

$= (5)_{10}$

* Decimal to Binary

① $(10)_{10} \rightarrow (1010)_2$

2	10	
2	5	0
2	2	1
	1	0

② $(15)_{10} \rightarrow (1111)_2$

2	15	
2	7	1
2	3	1
	1	1

32	16	8	4	2	1
	1	0	0	1	
	1	1	1	1	0
	0	1	0	0	0
	1	0	1	1	0
	0	1	0	0	1
	1	1	0	0	1
	1	0	1	1	

③ $(20)_{10} \rightarrow (10100)_{(2)}$

2	20	
2	10	0
2	5	0
2	2	1
	1	0

④ $(25)_{10} \rightarrow (11001)_2$

2	25	
2	12	1
2	6	0
2	3	0
	1	1

⑤ $(30)_{10} \rightarrow (11110)_2$

2	30	
2	15	0
2	7	1
2	3	1
	1	1

⑥ $(35)_{10} \rightarrow (100011)_2$

2	35	
2	17	1
2	8	1
2	4	0
2	2	0
	1	0

(7) $(40)_{10} \rightarrow (101000)_2$

2	40	
2	20	0
2	10	0
2	5	0
2	2	1
	1	0

(8) $(45)_{10} \rightarrow (101101)_2$

2	45	
2	22	1
2	11	0
2	5	1
2	2	1
	1	0

(9) $(50)_{10} \rightarrow (110010)_2$

2	50	
2	25	0
2	12	1
2	6	0
2	3	0
2	1	1
		0

(10) $(55)_{10} \rightarrow (100111)_{10}$

2	55	
2	27	1
2	13	1
2	6	1
2	3	0
	1	0

Decimal to octal

(1) $(945)_{10} \rightarrow (1661)_8$

8	945	
8	118	1
8	14	6
	1	6

(2) $(946)_{10} \rightarrow (2002)_8$

8	946	
8	118	2
8	16	0
	2	0

(3) $(182)_{10} \rightarrow (264)_{(8)}$

8	182	
8	21	4
	2	5

(4) $(256)_{10} \rightarrow (400)_8$

8	256	
2	32	0
	4	0

(5) $(645)_{10} \rightarrow$

8	645	
	85	

(6) $(924)_{10} \rightarrow (1634)_8$

8	924	
8	115	4
8	14	3
	1	6

(7) $(911)_{10} \rightarrow (1617)_8$

8	911	
8	113	7
8	14	1
	1	6

(8) $(852)_{10} \rightarrow (1404)_8$

8	852	
8	106	4
8	12	0
	1	4

(9) $(625)_{10} \rightarrow (1161)_8$

8	625	
8	78	1
8	9	6
	1	1

(10) $(749)_{10} \rightarrow (1355)_8$

8	749	
8	93	5
8	11	5
	1	3

* Octal to Decimal

(1) $(679)_8 \longrightarrow (449)_{10}$

$= 6\times8^2+7\times8^1+9\times8^0$

$= 6\times64+7\times8+9\times1$

$= 384+56+9$

$= (449)_{10}$

(2) $(528)_8 \longrightarrow (344)_{10}$

$= 5\times8^2+2\times8^1+8\times8^0$

$= 5\times64+2\times8+8\times1$

$= 320+16+8$

$= 344$

(3) $(496)_8 \longrightarrow (334)_{10}$

$= 4\times8^2+9\times8^1+6\times8^0$

$= 4\times64+9\times8+6\times1$

$= 256+72+6$

$= (334)_{10}$

(4) $(257)_8 \longrightarrow (175)_{10}$

$= 2\times8^2+5\times8^1+7\times8^0$

$= 2\times64+5\times8+7\times1$

$= 128+40+7$

$= (175)_{10}$

(5) $(349)_8 \longrightarrow (233)_{10}$

$= 3\times8^2+4\times8^1+9\times8^0$

$= 3\times64+4\times8+9\times1$

$= 192+32+9$

$= (233)_{10}$

(6) $(999)_8 \longrightarrow (657)_{10}$

$= 9\times8^2+9\times8^1+9\times8^0$

$= 9\times64+9\times8+9\times1$

$= 576+72+9$

$= (657)_{10}$

(7) $(964)_8 \longrightarrow (628)_{10}$

$= 9\times8^2+6\times8^1+4\times8^0$

$= 9\times64+6\times8+4\times1$

$= 576+48+4$

$= (628)_{10}$

(8) $(465)_8 \longrightarrow (309)_{10}$

$= 4\times8^2+6\times8^1+5\times8$

$= 4\times64+6\times8+5\times1$

$= 256+48+5$

$= (309)_{10}$

256
48
5
309

(9) $(648)_8 \longrightarrow (424)_{10}$

$6\times8^2+4\times8^1+8\times8^0$

$= 6\times64+4\times8+8\times1$

$= 384+32+8$

$= (424)_{10}$

(10) $(7014)_8 \longrightarrow (3596)_{10}$

$7\times8^3+0\times8^2+1\times8^1+4\times8^0$

$7\times512+0\times64+1\times8+4\times1$

$3584+0+8+4$

$= (3596)_{10}$

3596
384
32
8
424
6×8
3
512
3584
0

* Binary to octal

① $(0110)_2 \longrightarrow (6)_8$

$= 0\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0$

$= 0\times8 + 1\times4 + 1\times2 + 0\times1$

$= 0 + 4 + 2 + 0$

$= (6)_8$

② $(1100)_2 \longrightarrow (12)_8$

$= 1\times2^3 + 1\times2^2 + 0\times2^1 + 0\times2^0$

$= 1\times8 + 1\times4 + 0\times2 + 0\times1$

$= 8 + 4 + 0 + 0$

$= (12)_8$

③ $(110011)_2 \longrightarrow (51)_8$

$= 1\times2^5 + 1\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$

$= 1\times32 + 1\times16 + 0\times8 + 0\times4 + 1\times2 + 1\times1$

$= 32 + 16 + 0 + 0 + 2 + 1$

$= (51)_8$

④ $(11010)_2 \longrightarrow (26)_8$

$= 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$

$= 1\times16 + 1\times8 + 0\times4 + 1\times2 + 0\times1$

$= 16 + 8 + 0 + 2 + 0$

$= (26)_8$

⑤ $(1110)_2 \longrightarrow (14)_8$

$= 1\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0$

$= 1\times8 + 1\times4 + 1\times2 + 0\times1$

$= 8 + 4 + 2 + 0$

$= (14)_8$

⑥ $(1111)_2 \longrightarrow (15)_2$

$= 1\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0$

$= 1\times8 + 1\times4 + 1\times2 + 1\times1$

$= 8 + 4 + 2 + 1$

$= (15)_2$

⑦ $(10001) \longrightarrow (17)_8$

$= 1\times2^4 + 0\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0$

$= 1\times16 + 0\times8 + 0\times4 + 0\times2 + 1\times1$

$= 16 + 0 + 0 + 0 + 1$

$= (17)_8$

⑧ $(101101) \longrightarrow$

$= 1\times2^5 + 0\times2^4 + 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$

$=$

B - D $\rightarrow \times 2$

D - B $\rightarrow \div 2$

B - O $\rightarrow \times 2$

O - B $\rightarrow \times 8, \div 2$

D - O $\rightarrow \div 8$

O - D $\rightarrow \times 8$

v Hexa decimal conversion table

1
2
3
4
5
6
7
8
9
10 → A
11 → B
12 → C
13 → D
14 → E
15 → F

Hexadecimal to decimal

* $(62C)_{16} \Rightarrow$

C = 12 $6 \times 16^2 + 2 \times 16^1 + 12 \times 16^0$

$= 1536 + 32 + 12$

$=$

* $(AB12)_{16} \rightarrow (43796)_{10}$

~~A × 16 + 11 ×~~

$10 \times 16^3 + 11 \times 16^2 + 1 \times 16^1 \times 2 \times 16^0$

~~10 × 256 + 11 ×~~

$= 10 \times 4096 + 11 \times 256 + 1 \times 16 + 2 \times 1$

$= 40960 + 2816 + 16 + 2$

the $= 43796$

* $(8ED4)_{16} \rightarrow (36532)_{10}$

$8 \times 16^3 + 14 \times 16^2 + 13 \times 16^1 + 4 \times 16^0$

$= 8 \times 16 \times 16 + 14 \times 16 \times 16 + 13 \times 16 + 4 \times 16$

$= 8 \times 4096 + 14 \times 256 + 13 \times 16$

$= 32768 + 3584 + 209 + 4$

$= \cancel{38}\ (36532)_{10}$

32768
3584
209
4

8

$(A7B)_{16} \rightarrow (2683)_{10}$

$10 \times 16^2 + 7 \times 16^1 + 11 \times 16^0$

$= 10 \times 256 + 7 \times 16 + 11 \times 1$

$= 2560 + 112 + 11$

$= (2683)_{10}$

$= 10 \times 256 \times 10$

000
256

2560
112
11
2683

16

* Hexadecimal to Binary

* $(B2E)_{16} \rightarrow (1011\ 0010\ 1110)_2$

B = 11 ⟹ 1011
2 = 2 ⟹ 0010
E = 14 ⟹ 1110

* $(A3C)_{16} \rightarrow (1010\ 0011\ 1100)_2$

A = 10 = 1010
3 = 2 = 0011
C = 12 = 1100

* Hexadecimal to octal

* $(B2E)_{16} \rightarrow$ ~~(1011 0010 1110)~~ $(5456)_8$

B → 11 = 1011
2 → 2 = 0010
E → 14 = 1110
~~1011 0010 1110~~

101100101110
5 4 5 6

* $(A7B)_{16} \rightarrow (5173)_8$

A = 10 = 1010
7 = 7 = 0111
B = 11 = 1011
~~1010~~

101001111011
5 1 7 3

* Decimal to Hexadecimal

$(243)_{10} \rightarrow (F3)_{16}$

16	243	16	243	
	15		15	3

* $(624)_{10} \rightarrow$

16	624	
16	39	0
	2	7

1A9

+ $(157)_{16} \rightarrow$

798.

num = 789

rev n = 0.

rem = n % 10

= 789 % 10.

rem. = 9

revn = revn * 10 + r

= 0 * 10 + 9

= 9.

num = 78.

rev n = 9.

rem = n % 10

= 78 % 10

rem = 8

rev = revn * 10 + r

= 9 * 10 + 8

= 90 + 8

= 98

num = 7.

rev n = 98.

rem = n % 10

= 7 % 10

rem = 7

rev = rev * 10 + r

= 98 * 10 + 7

= 980 + 7

rev n = 987

Reversed number. write Algo and flow

Reversed num = 0

rem = num % 10
rev = rev * 10 + rem
num = num / 10

① n = 321
= rev = 0
rem = 321 % 10
rem = 1
rev

rev = rev * 10 + rem
= 0 * 10 + 1
= 0 + 1
= 1

num = num / 10
= 3

② n = 32
rev = 1
rem = 32 % 10
= 2
rev = rev * 10 + rem
= 1 * 10 + 2
= 10 + 3
= 12

③ n = 3
= rev = 12
rem = 3 / 10
= 3
rev = rev * 10 + rem
= 12 * 10 + 3
= 120 + 3
= 123

② n = 113
rev = 0
rem = 113 % 10
rem = 3
rev = rev * 10 + rem
= 0 * 10 + 3
= 0 + 3
= 3

② n = 11
rev = 3
rem = 11 % 10
= 1
rev = rev * 10 + rem
= 3 * 10 + 1
= 30 + 1
= 31

③ n = 1
rev = 31
rem = 1 % 10
rem = 1

rev = rev * 10 + rem
= 31 * 10 + 1
= 310 + 1 = 311

31 × 10
00
31

31 × 10
00

* Reversed number

```
#include<stdio.h>
int main()
{
  int num, rev=0, rem;
  printf("Enter the number");
  scanf("%d", &num);
  while(num!=0)
  {
    rem= num%10;
    rev = rev * 10 + rem;
    num= num / 10;
  }
  printf("reversed number =%d", rev);
}
```

Algorithm :-

Step 01 :- Start

Step 02 :- Read num

Step 03 :- Cheak num != 0

Step 04 :- Compute rem=num%10

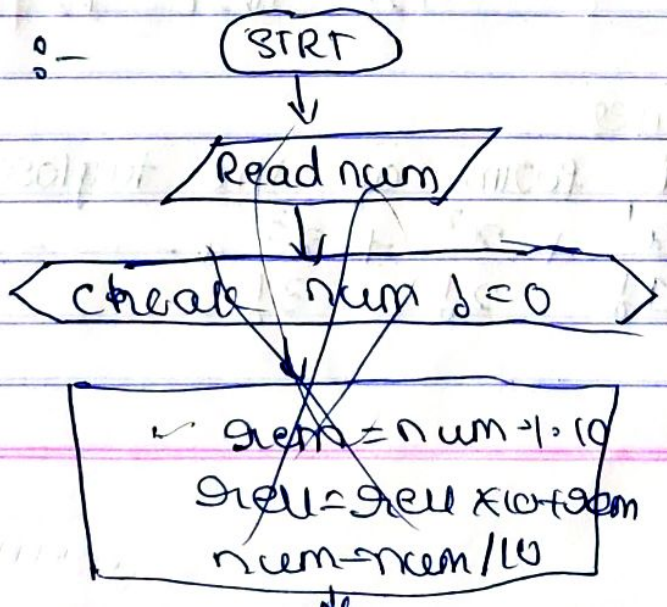
rev=rev*10 + rem

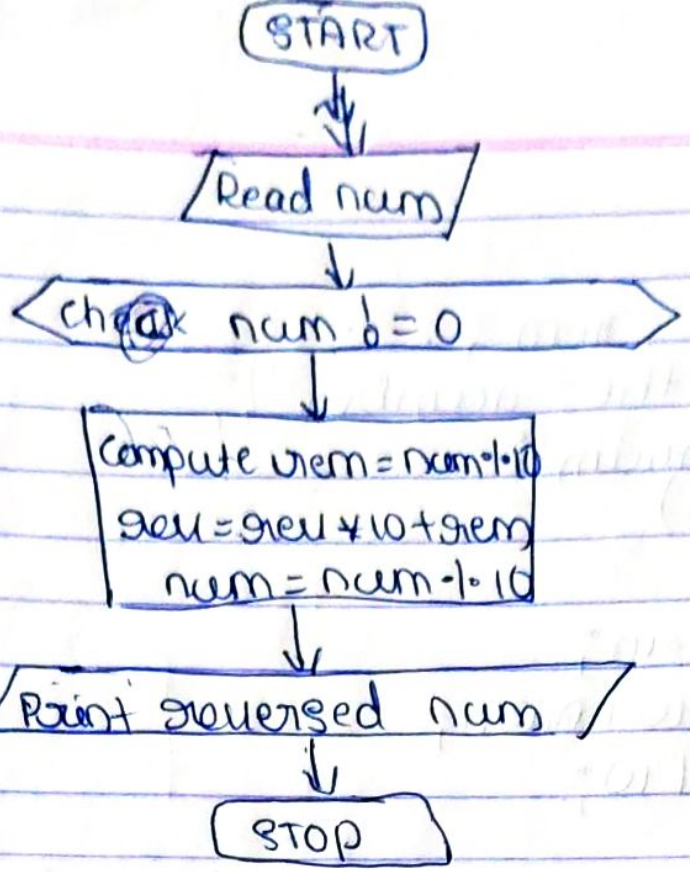
num= num / 10;

Step 05 :- print reversed number =%d, rev

Step 06 :- End.

flow Chart :-





Sine table

~~function~~	degree		
~~Sine~~	0		
~~Sin~~	0	30°	45°
sine	0	$\pi/6$	
cos	0	$1/2$	

~~table~~

~~function~~ table

	0°	30°	45°	60°	90°
sine	0	$\pi/6$ → radian	$\pi/4$	$\pi/3$	$\pi/2$
	0	$1/2$	$1/\sqrt{2}$	$\sqrt{3}/2$	1
cos	1	$\sqrt{3}/2$	$1/\sqrt{2}$	$1/2$	0

Taylor's Series

The general form of the taylor's series

$$x \text{ or } x(n) = x + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$