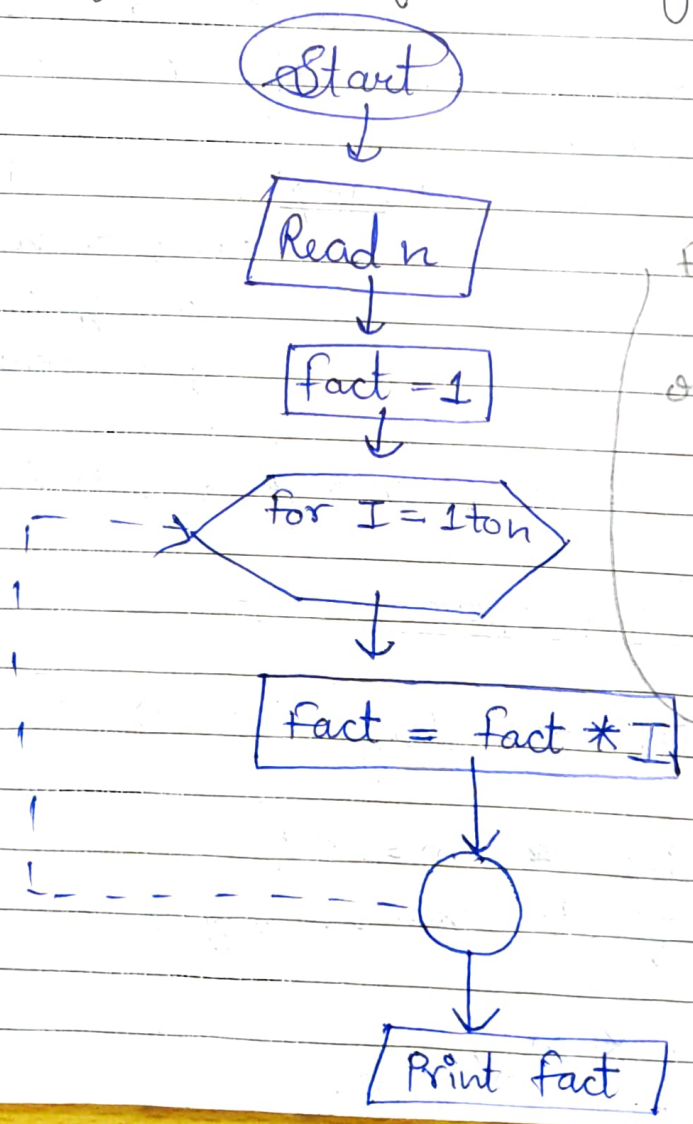
(2) int m = 5;
float y;

y = m/2;

y = (float) m/2;

Algorithm

(1) Draw a flow chart to compute the factorial of a given number

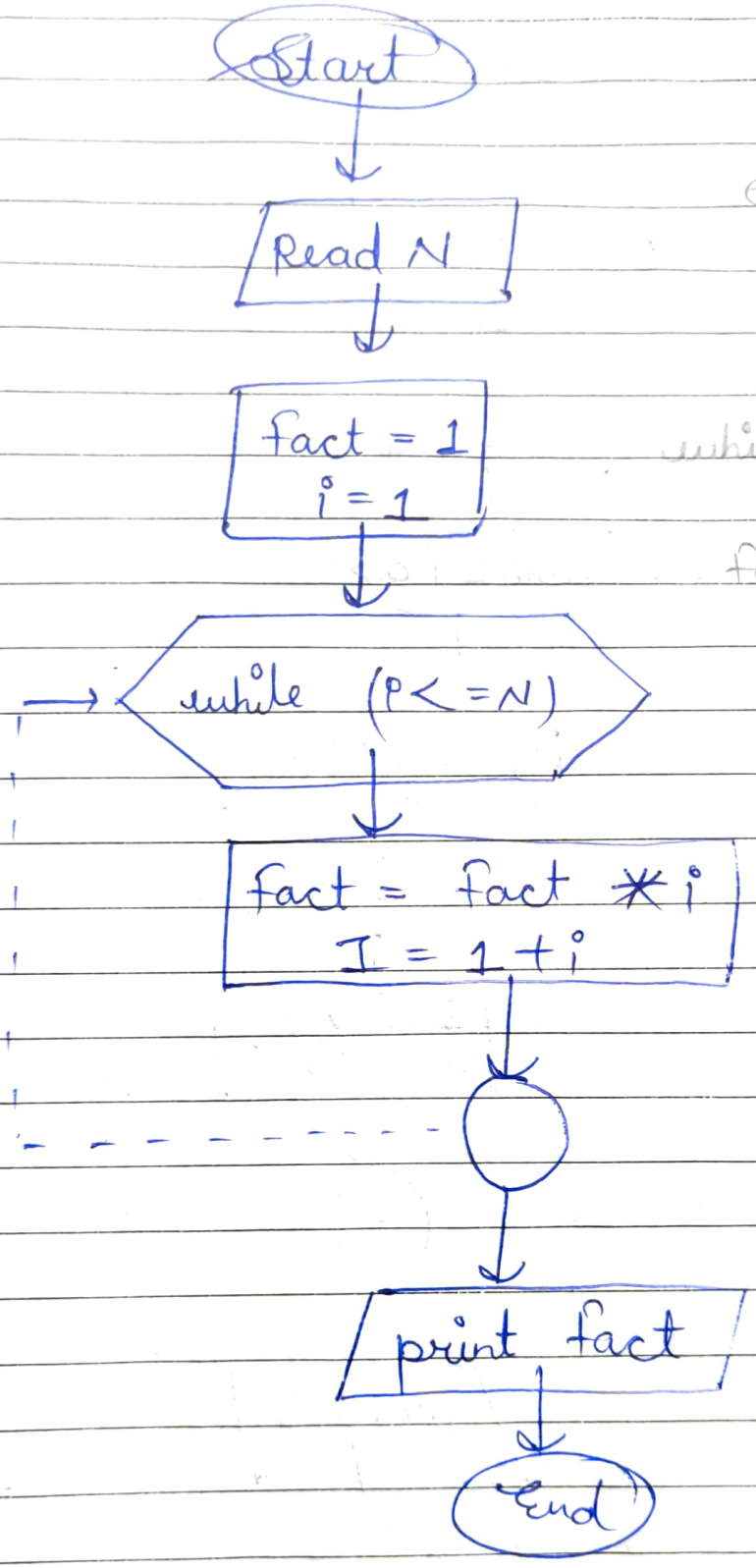


Eg :- n = 5
fact = 1
for I = 1 to 5
fact = fact * 1 to 5
= 5 x 4 x 3 x 2 x 1
output = 120.

1 * 1
1 * 2
1 * 2 * 3
1 * 2 * 3 *
1 * 2 * 3 *

= 120

OR



eg :-

N = 5

fact = 1

i = 1 to 5

while (1 ≤ 5)

i = 1+1 = 2

fact = 1*1 = 1

(2 ≤ 5)

fact = 1*2 = 2

i = 2+1 = 3

(3 ≤ 5)

fact = 1*2*3 = 6

i = 3+1 = 4

(4 ≤ 5)

fact = 4*5 = 24

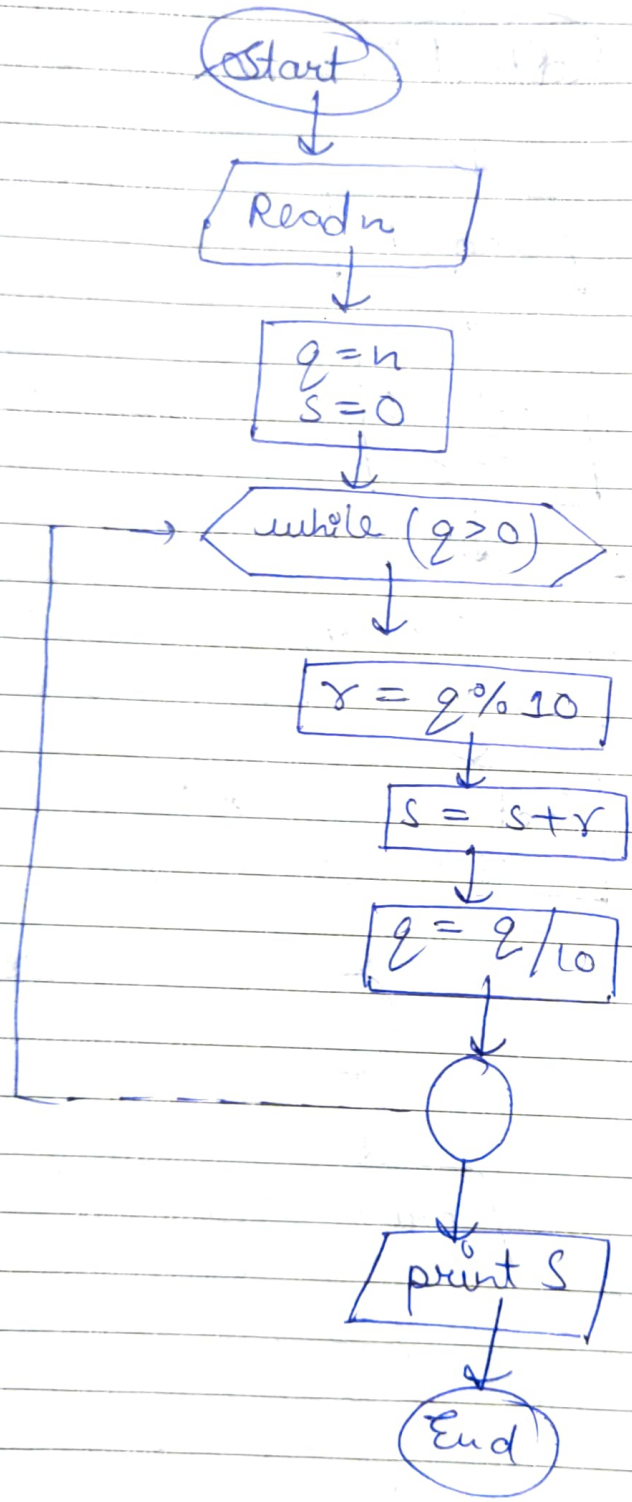
i = 4+1 = 5

(5 ≤ 5)

fact = 24*5 = 120

i = 5+1 = 6

Draw a flow chart for sum of Numbers.



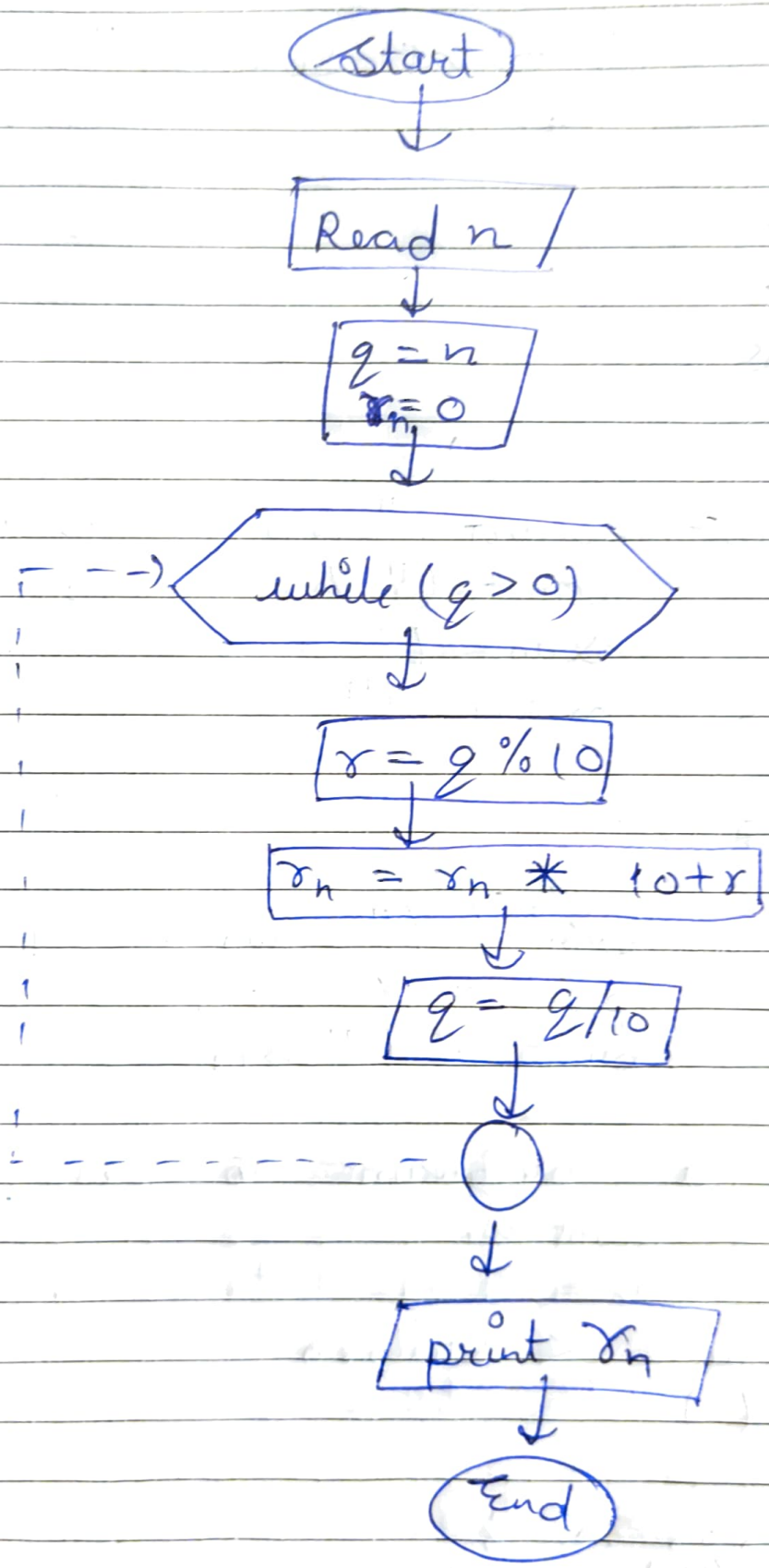
e.g. n = 123
q = 123
s = 0
while (123 > 0)
r = 123 % 10 = 3
s = 0 + 3
s = 3
q = 123/10 = 12

while (12 > 0)
r = 12 % 10 = 2
s = 0 + 2
s = 2
q = 12/10 = 1

while (1 > 0)
r = 1 % 10 = 1
s = 0 + 1
s = 1
q = 1/10 = 0

output = 3 + 2
= 6.

Reverse process :- 321 → 123



e.g. n = 321
q = 321
rn = 0
while (321 > 0)
r = 321 % 10 = 1
rn = 0 * 10 + 1
= 0 + 1 = 1
321 = 321/10 = 32

32 > 0
r = 32 % 10 = 2
rn = 1 * 10 + 2
rn = 12
q = 32/10 = 3

3 > 0
r = 3 % 10 = 3
rn = 12 * 10 + 3
rn = 123
q = 3/10 = Not exceed

output :-
123

Scanf :-

```
Scanf (" format specifier", &V1,
        &V2 ------ &V3);
```

%d
%c
%s
%f

Example :-

```
Scanf ("%d", &a);
Scanf ("%c", &choice);
Scanf ("%s", &name);
Scanf ("%f", &r);
```

Print f :-

Example :-

```
print f (" Sum = %d ", s);
print f (" Sum = ");
print f (" Sm = ");
```

Q. Write a program to print the sum of two numbers.

```
# include < stdio.h >
main ( )
{
int a, b, s ;
print f ("\n Enter the value of a");
Scan f ("%d %d", &a, &b)
```

or

```
scanf ("%d", &a);
printf ("\n Enter the value of b");
scanf ("%d", &b);
s = a+b;
Printf ("\n sum = %d", s);
}
```

<u>formatted output function</u> :-

```
int n = 28;
Printf ("%5d", n);
```

b	b	b	2	8

```
Printf ("%-5d", n);
```

2	8	b	b	b

```
Printf ("%+5d", n);
```

b	b	+	2	8

```
float x = 28.358416
print f ("%f", x);
```

2	8	.	3	5	8	4	1	6

Print f ("% 6.2f", x);

~~b~~	2	8	.	3	6

Print F ("% 0.2f", x);

2	8	.	3	6

Print f ("% 0.3 e", x);

2	.	8	4	e	+	0	1

Char st [10] = "UDAIPUR"

Print f ("% s", st);

U	D	A	I	P	U	R

Print f ("% 10 s", st);

~~b~~	~~b~~	~~b~~	U	D	A	I	P	U	R

Print f ("% 10.3 s", st);

~~b~~	~~b~~	~~b~~	~~b~~	~~b~~	~~b~~	~~b~~	U	D	A

Print f ("% - 10.3 s" st);

| U | D | A | ~~b~~ | ~~b~~ | ~~b~~ | ~~b~~ | ~~b~~ | ~~b~~ | ~~b~~ |

Print f ("% 0.3 s", st);

| U | D | A |

Character - input - output function :-

get char ( ) :- Read 1 character at a time with standard input device

Syntax :-

```
ch = get char ( );
```

where ch is a character variable

Eg :-

```
Char ch;
ch = get char ( );
```

put char ( ) :- Use to display 1 character at a time it has the following form

Syntax :- put char (ch);

E.g. :-

```
char ch = 'M';
put char (ch);
```

getch ( ) :- Use to read a character from the keyboard and it does not espect the enter key press.

Syntax :- ch = getch ( );

where ch is a char variable

```
char c;
```

Eg :-

```
c = getch();
```

The computer wait for a key to be pressed from the keyboard as scon as the key is pressed. The control is transfered to the next line of the program and the value is assigned to the char variable

Putch () :- used to display a character on the monitor.

The syntax is putch (ch); where 'ch' is a char variable

```
char c = 's';
putch (c);
```

get s () :- It is used to read a string of characters including wide space. A string containing wide space can not be read using scanf function with % s format specifier.

The Syntax is :-

```
get s (st);
```

<u>puts ( )</u> :- It is used to display a character string on the monitor it has the following form :-

```
puts (str);
```

where str is a string.

<u>Eg</u> :- char st [20] = " Computer Engg";

So here if we are using

```
puts (st);
```

it will print the value of st.

<u>Control statement</u> :- It is used to execute / transfer the control from one part of the program to another depending on a condition.
The statement is also called as conditional statement.

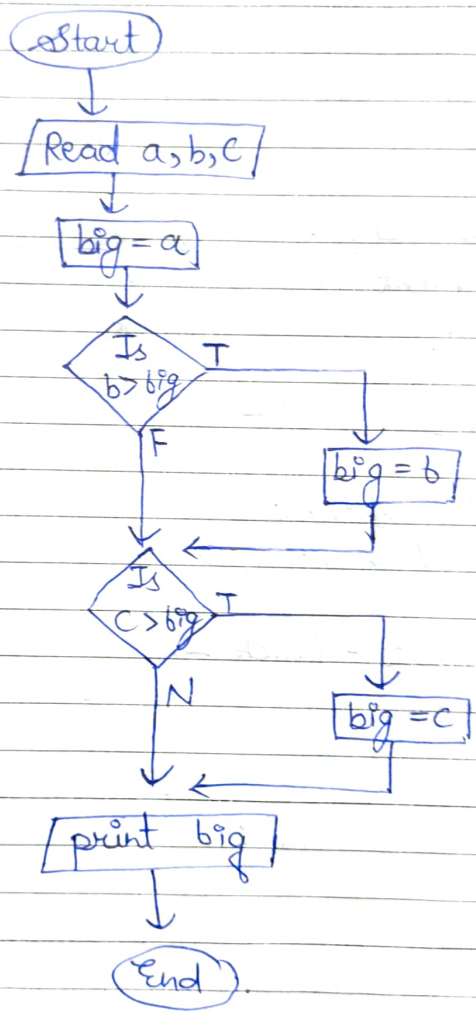
(1) <u>If - else statement</u> :- used to execute a statement block or a single statement depending on the value of a condition

<u>Syntax</u> :-

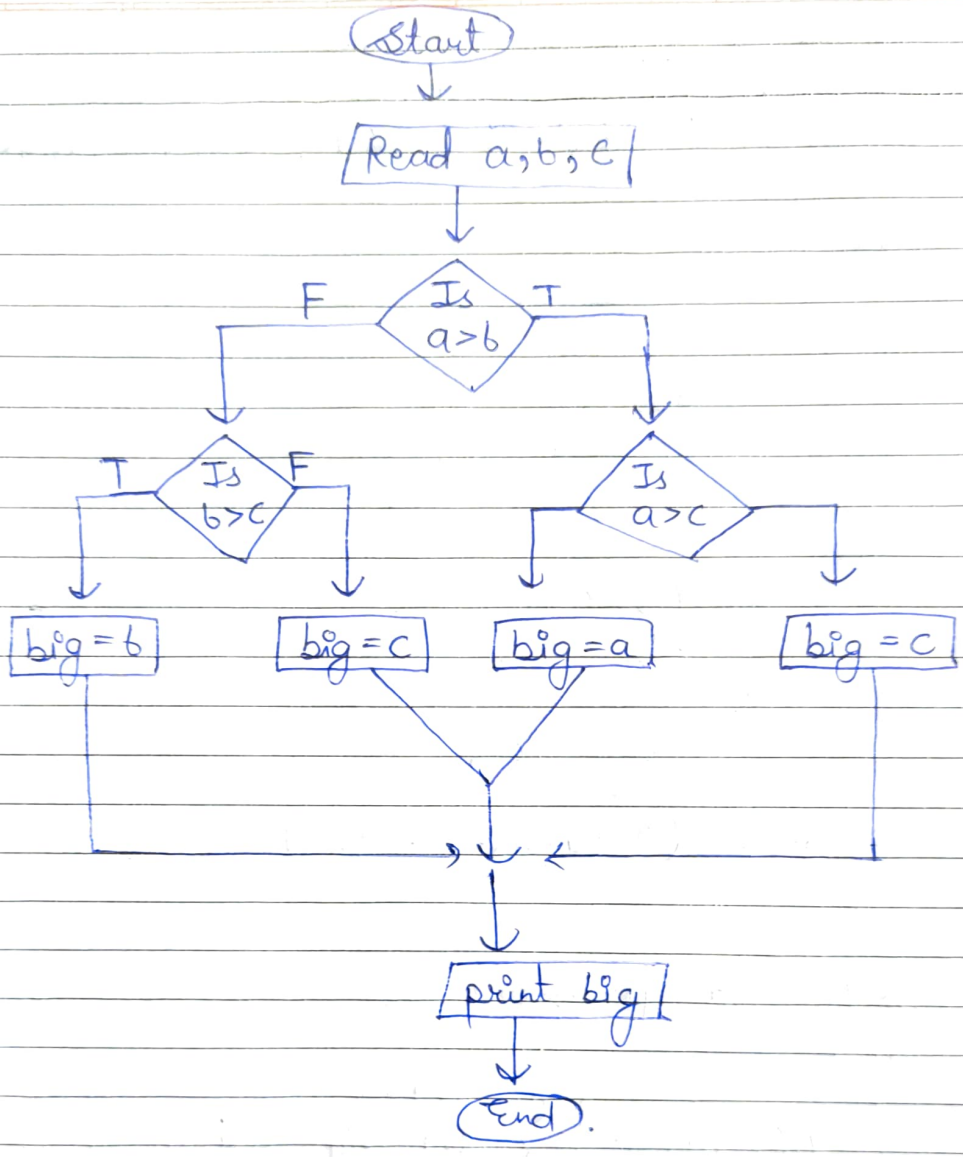
```
if (condition)
  {---------
```

Example :-

→ Algorithm for largest no. among three numbers :-



OR



Program :- for the same algorithm.

```
#.......
{ main ( )
  int a, b, c , big ;
```

```
Print f ("\n Enter a, b, c");
Scan f ("%d %d %d, &a, &b, &c);

if (a>b)
        if (a>c)
                big = a;
        else
                big = c;

else
        if (b>c)
                big = b;
        else
                big = c;

Print f (" Largest = % d", big);
}
```

Q. Write a program to find the value of y.

$$y(x,n) = \begin{cases} 1+x & \text{when } n=1 \\ 1+x/n & \text{when } n=2 \\ 1+x^n & \text{when } n>3 \\ 1+nx & \text{when } n>3 \text{ or } n<1 \end{cases}$$

Just like :-

$n=3 \Rightarrow y = 1+x^n$

$n=1 \Rightarrow y = 1+x$

$n=0 \Rightarrow y = 1+nx$

$n=-1 \Rightarrow y = 1+nx$

$n=5 \Rightarrow y = 1+nx$

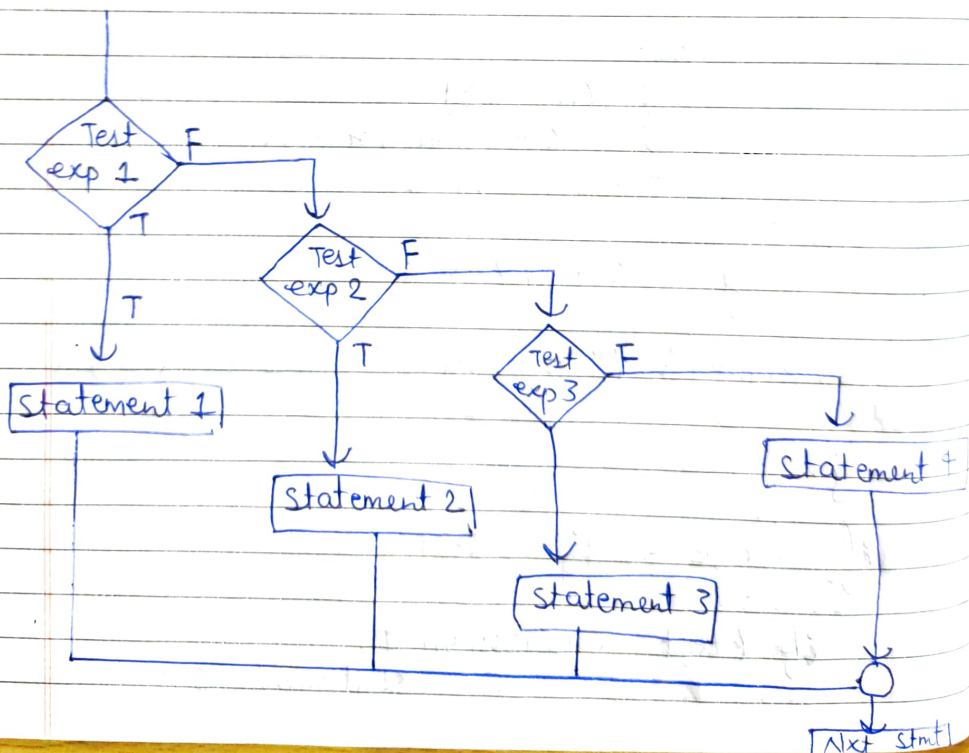
Program execution :-

```
# include < stdio.h >
# include < math.h >
main ( )
{
 int n;
 float x, y;
 Print f ("\n Enter x & n");
 scanf (" %f %d ", &x , &n);
 if (n==1)
        y = 1+x;
    else
      if (n==2)
      y = 1 + x/n ;

    else
      if (n==3)
      y = 1+ pow (x,n);

else
      y = 1+ n*x;

 Print f ("\n  y= % 6.2 f", y);

}.
```

if - else - if ladder :-

```
if (test expression 1)
                statement 1;
```

```
else if (test - expression 2)
                    statement 2;

else if (test - expression 3)
                    statement 3;
        |
        |
        |
        |
        |
else
        statement;
```



Example :- If the :-

Avg marks	Grade
80 to 100	Honour
60 to 79	First
50 to 59	Second
0 to 49	Fail

```
if (Avg. marks > 80 && Avg. marks < 100)
        Print f ("\n Honour");

    else
        if (Avg. marks > 60 && Avg. marks < 79)
            print f ("\n First");

    else
        if (Avg. marks > 50 && Avg. marks < 59)
            print f ("\n Second");

else
    print f ("\n Fail");
```

Question :- Check the given number is +ve, -ve and 0.

```
main ()
{
  Int x;
Print f ("\n Enter the number");
Scan f ("%d", &x);
```

```
if (x > 0)
        printf("\n Given number is positive");
else
    if (x == 0)
        printf ("\n Given number is zero");
else
        printf ("\n Given number is negative");
}
```

* **Switch Statement** :- It is used to execute a block of statement depending on the value of a variable or an expression

Syntax :-

```
Switch (<Expression>)
{
  Case <label 1> : {
                    <statement block 1>
                    break;
                    }
  Case <label 2> : {
                    <statement block 2>
                    break;
                    }
  Case <label 3> : {
                    <statement block 3>
                    break;
                    }
```

```
case < label n > : {
                   < statement - block n >
                   break ;
                   }

default : {
          < default - statement - block >
          break ;
          }

}
```

where expression refers any int or char expression or variable. Label 1, label 2, --- --- label n are the values which will match with the value of expression.

Note :-

1. The braces { } can be omitted when there is only one statement available in the statement block.

2. The break statement is used to transfer the control to the end of switch statement.

3. The default block is executed when non of the case labels match with the value of the expression / variable.

4. The default block is optional, these may be omitted while writing a program.

Switch statement is alternate of ladder if - else statement.

<u>Example</u> :- $y(x,n) = 1+x$, when $n=1$
$1+\frac{x}{n}$, when $n=2$

$1+x^n$, when $n=3$
$1+nx$, when $n>3$ or $n<$

```
main ( )
{
  int n;
  float x, y;

      Printf ("\n Enter the value of x, n");
      scanf ("%f %d", &x, &n);

    switch (n)
    {
     case 1 :
              y = 1+x;
              break;

     Case 2 :
              y = 1+ x/n;
              break;

     case 3 :
              y = 1 + pow(x,n);
              break;
```

```
default :
        y = 1 + n * x
        break ;
    print f ("\n y = % 6.2 f", y);
}
```

Q.

MENU
Enter first No :
Enter second no :
Select the operation
1. Addition
2. Substraction
3. Multiplication
4. Division
Enter your choice : _____
The Result is _____

Write the Syntax of given program

Syntax :-

```
main ( )
{
 int a, b, ch ;
```

PTO

```
Print f ("\n \t MENU \ n");
Print f ("\n Enter first number");
scanf ("%d", &a);
Print f ("\n Enter second number");
scanf ("%d", &b);
Print f ("\n \t select the option \n");
Print f ("\n 1. Addition");
Print f ("\n 2. substraction");
Print f ("\n 3. Multiplication");
Print f ("\n 4. Division");
Print f ("\n Enter your choice");
scanf ("%d", &ch);

    Switch (ch)
  {
   case 1 :
            Print f ("\n The result is %d",
                                    a+b);
            break;
 case 2 :
          Print f ("\n The result is %d", a-b);
          break;
 case 3 :
          Print f ("\n The result is %d", a*b);
 case 4 :
          Print f ("\n The result is %d", a/b);
 default :
          Print ("\n wrong choice! Retry again")
          break;
}  }
```

Q. Write a program to check the given year is leap year.

The condition is :-

The year is divisible by 4 but not divisible by 100 or year divisible by 400.

```
main()
{
  int year, r_4, r_100, r_400;
  Printf("\n Enter a year");
  Scanf("%d", &year);
  r_4 = year % 4;
  r_100 = year % 100;
  r_400 = year % 400;
if ((r_4 == 0 && r_100 != 0) || (r_400 == 0))

      Printf("\n Given year %d is leap year",
                                  year);
else
      Printf("\n Given year %d is not leap
                                  year", year);

}
```

<u>for loop</u> :-

A for loop is used to execute and repeat a block of statement depending on a condition, it has the following form :-

```
for(< initial value> ; < condition >; increment)
{ - - - - - - - - - -
  < Statement - block >
  - - - - - - - - - -
}
```

Here, initial value is the assignment expression which initialize the value of the variable. condition is the relational or logical expression, which will have the value True / False.
Increament is the increament value of the variable which will be added every time.

<u>Example</u> :-

```
for (i=1 ; i<=10 ; i+1)
{                          let s=0, p=0 , i=1
    s = s + i ;            s = 0+1 = 1    s = 1+1 =
    p = p * i ;            p = 0*1 = 0    p = 0*1
}
```

doubt ?

When this statement is executed the computer assign initial value to variable and the condition is evaluated. If the value of the condition is True statement block will be executed the value of the variable is incremented and the condition is evaluated and is repeated until

the value of condition is false.

Note :-

(1) The braces ({}) can be omitted when there is only one statement available in the statement - block.

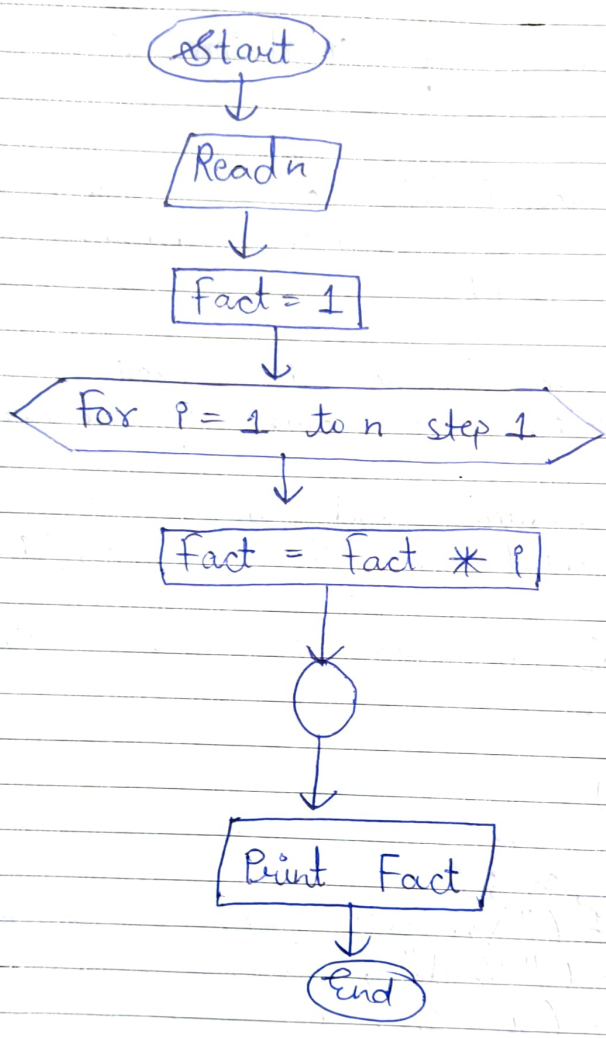
(2) The initial value is executed only one (1) to initialize the value of variable.

(3) The condition and increament are evaluated and executed on its iteration / Repetation.

Q. Write C program to print natural number 1 to n.

```
main ( )
{
  int i, n ;
  Print f ("\n Enter the value of n");
  Scanf ("%d", &n);
      for (i = 1 ; i <= n ; i + 1)

                Print f ("%5d", i);
}
```

Q Write a C program to find the factorial of a given number




```
main ( )
{
    int i, n , fact = 1 ;
    Print f ("\n Enter the value of n");
    Scan f ("%d", &n);

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