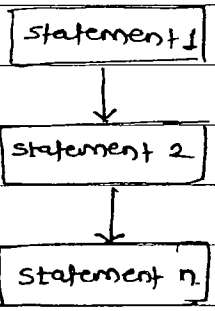
Control Flow

Program is a collection of instructions. Any program can be Construct (build) using sequential instructions Conditional instructions or looping instructions. These are the three types of programming instructions. In any program, instructions may be executed sequentially, selectively or iteratively. So every programming language provides following types of programming instructions (statements)

① Sequential statements
② Selective / Conditional statements
③ Iterative / Looping statements.

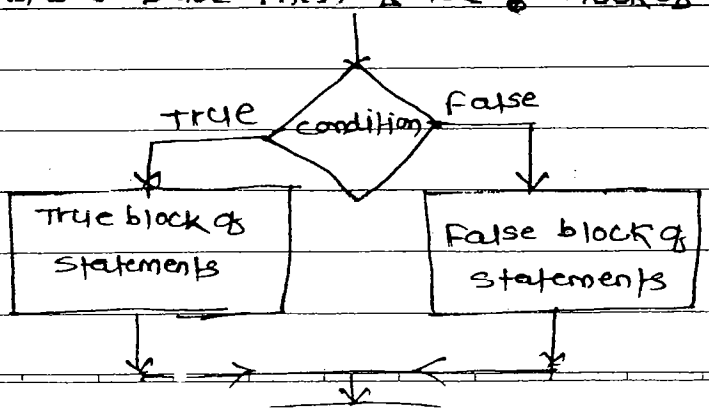
① Sequential statements :-

Sequential statements means instructions in program can be executed sequentially i.e. one after another.



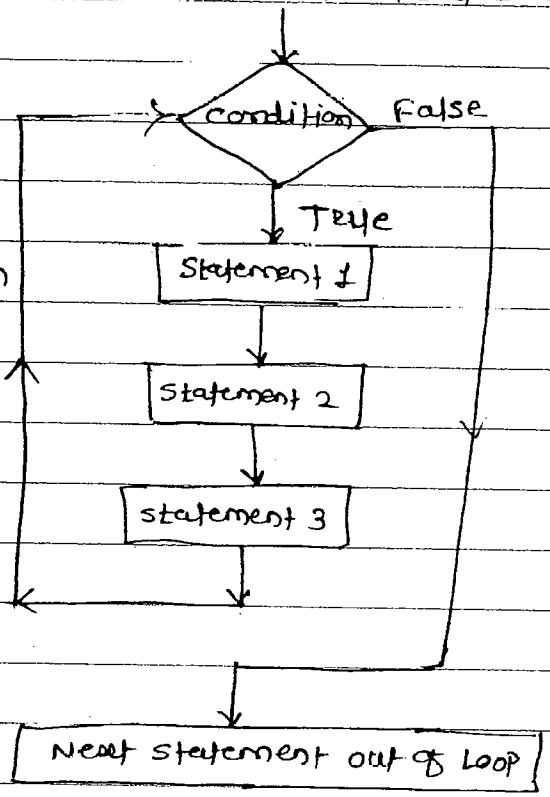
② Selective / Conditional Statements :-

Conditional Statements means execution of instruction is depending upon the condition. If condition evaluates to true then true block of statements get execute, if condition evaluates to false then false block of statements get execute



③ Iterative / Looping Statements :-

Iterative Statements means repetition of set of instructions depending upon a condition. when condition evaluates to true set of instructions are repeated again & again till condition evaluates to false.



* Statement:-

Statement in program is nothing but any instruction. Instruction is any single line in a program, through which we can give command to computer to perform certain execution.

* Block:-

Block is a set of instructions. Sometime in a program instructions can be used in group, so when instruction can be grouped together to perform certain operation then it is called as block. In C programming the block can shown by ({ , }). curly brackets.

* Conditional statements :-

In C programming following are the conditional statements.

1. Simple - if
2. if - else
3. Nested if - else
   - using if-else
   - using else if ladder

(1) Simple if statement :-

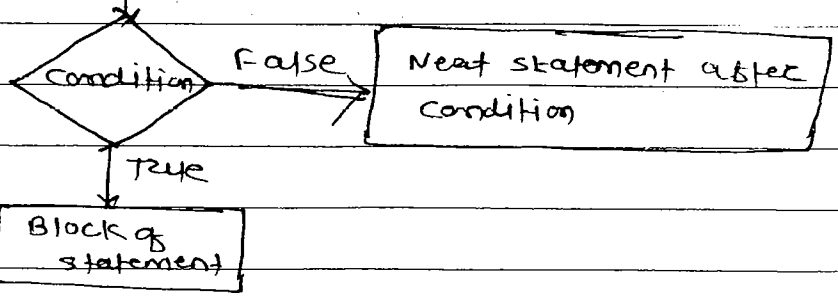
The simple if is used to test condition & execute statements depending upon the result of condition. The simple if statement is used only when we want condition should be evaluated to true.

Syntax :-

```
if (Condition)
{
    ---------
    ---------   Block of statements
    ---------
}
Next statement
```

As shown in syntax if is a keyword that tells to processor to check the condition, if condition is true then execute given block of statements, when condition evaluates to false processor does not execute block of statements, it will jump to next statement after if condition.

Flowchart



Program for simple if

```
#include<stdio.h>
void main()
{
int age;
Printf("\n Enter your age\n");
Scanf("%d",&age);
        if(age>=18)
        {
         Printf(" you are elligible for voting\n");
        }
}
```

As shown in above program, when condition evaluates to true then only printf statement get execute, if condition evaluates to false nothing will be executed. Here in program no any next instruction is there, but some time in other program any instruction will be there after block of statements.

* Write a program to show student is pass

```
#include<stdio.h>
void main()
{
int marks;
Printf("\n Enter your marks\n");
Scanf("%d",&marks);
        if(marks>=35)
        {
         Printf("student is pass\n");
        }
}
```

(2) if ... else Statement :-

In C if-else is a powerful statement. In simp if we can execute true block of statement, but we can't execute false block of statements when condition is false.

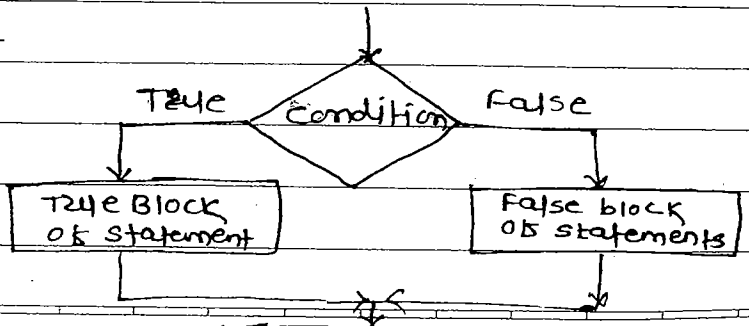
The if-else is extention of simple if statement, in which we can execute the true block of statements when condition is true & we can execute false block of statements when condition is false. In if-else we can get only one output either true or false depending on condition.

Syntax :-

```
if (Condition)
{
    - - - - -
    - - - - -    True block
    - - - - -    of statements
    - - - - -
}
else
{
    - - - - -
    - - - - -    False block
    - - - - -    of statements.
    - - - - -
}
Next statement;
```

As shown in syntax first processor checks the condition if condition is true then true block get execute false will skip & if condition is false then false block get execute true block will skip.

Flowchart



Program

```
#include<stdio.h>
void main()
{
int age;
Printf("\n Enter age\n");
Scanf("%d", &age);
        if(age>=18)
        {
         Printf("\n You are elligible for voting\n");
        }
       else
        {
          Printf("\n You are not elligible for voting\n");
        }
}
```

```
# include<stdio.h>
void main()
{
int n;
Printf("\n Enter Number\n");
Scanf("%d", &n);
            if(n%2==0)
            Printf("\n Number is even\n");
            else
            Printf("\n Number is odd\n");
}
```

As shown in above program there is no use of curly brackets ({, }) in if-else because there is only one instruction inside true block & false block. Curly brackes are optional when only one instruction inside true block & false block, but

- one instruction inside true block & false block of condition.

③ Nested if .... else

Nesting of condition is used when there is requirement of more than one condition in program. Nesting of condition is nothing but putting one condition inside block of another condition.

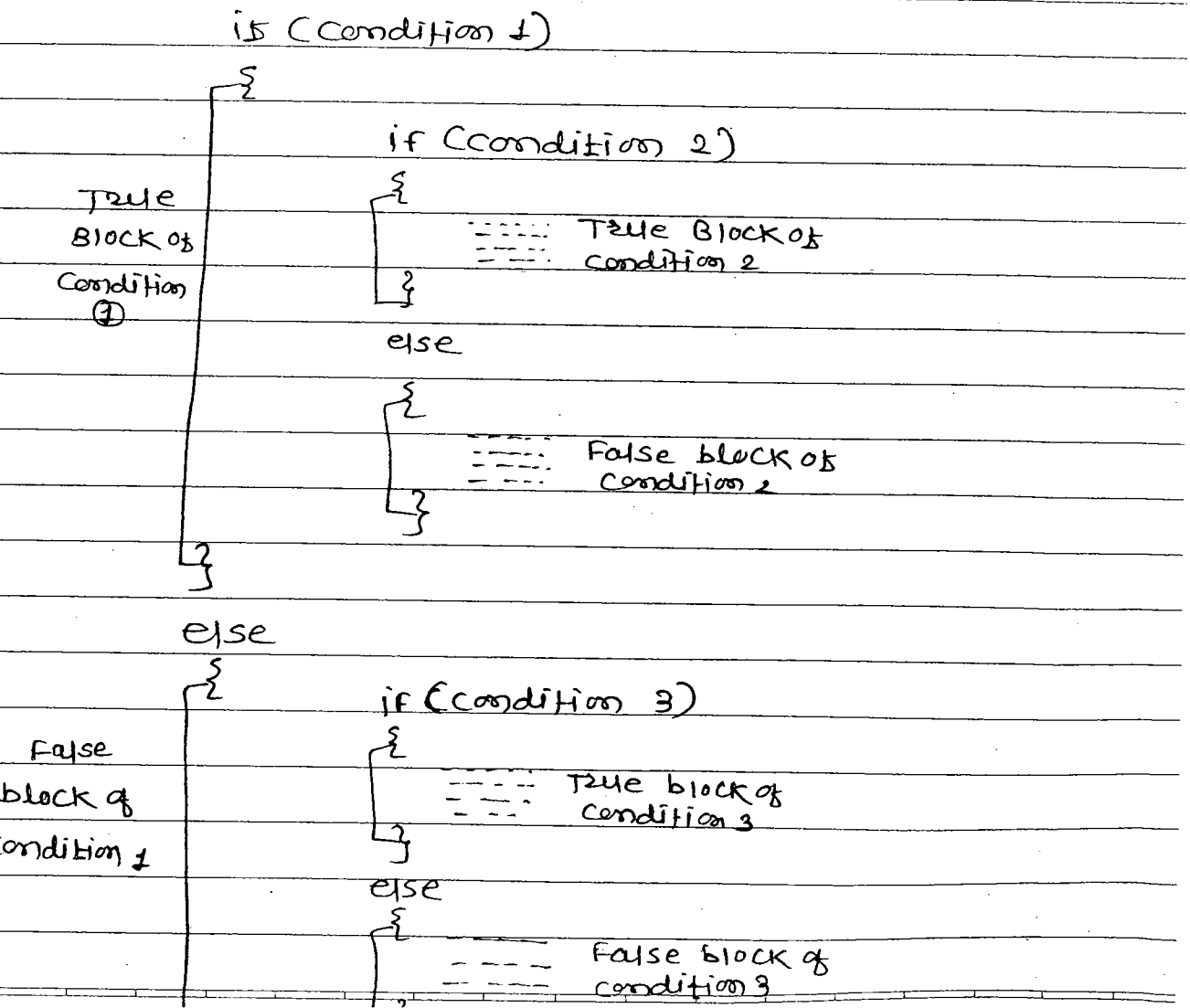
There are two types of Nesting of condition

① using if ... else

② using else if ladder.

① Using if- else

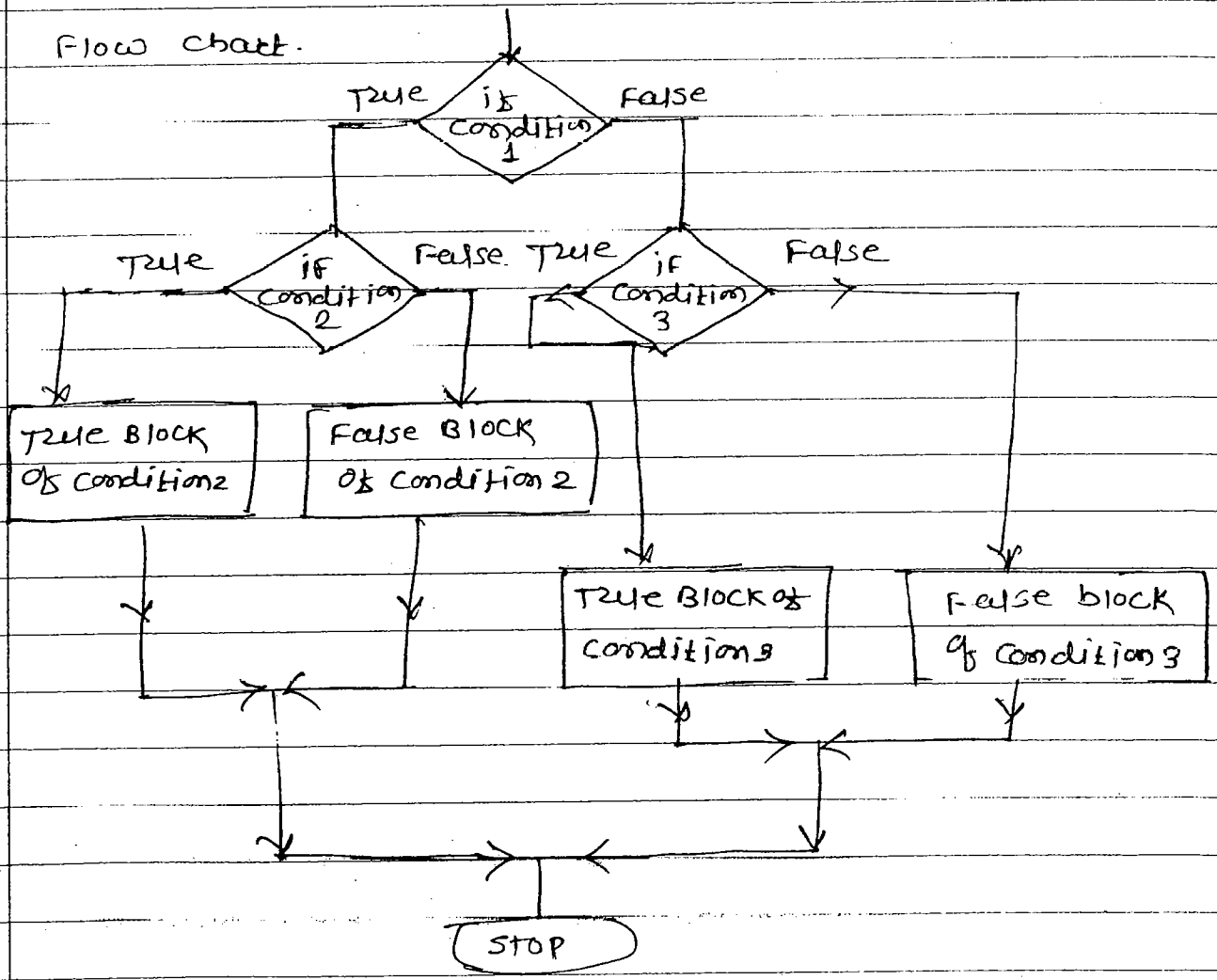
syntax:-



In nested if else, as shown in above syntax firstly processor checks the first condition if condition 1 is true then executes true block of condition 1 i.e. condition 2 cause condition 2 is a true block of condition 1, if the condition 2 is true then it execute true block of condition 2, if condition 2 is false then it executes false block of condition 2.

If condition 1 is false then it executes false block of condition 1 i.e. condition 3 cause condition 3 is a false block of condition 1, if condition 3 is true then it execute true block of condition 3, if condition 3 is false then it executes false block of condition 3.

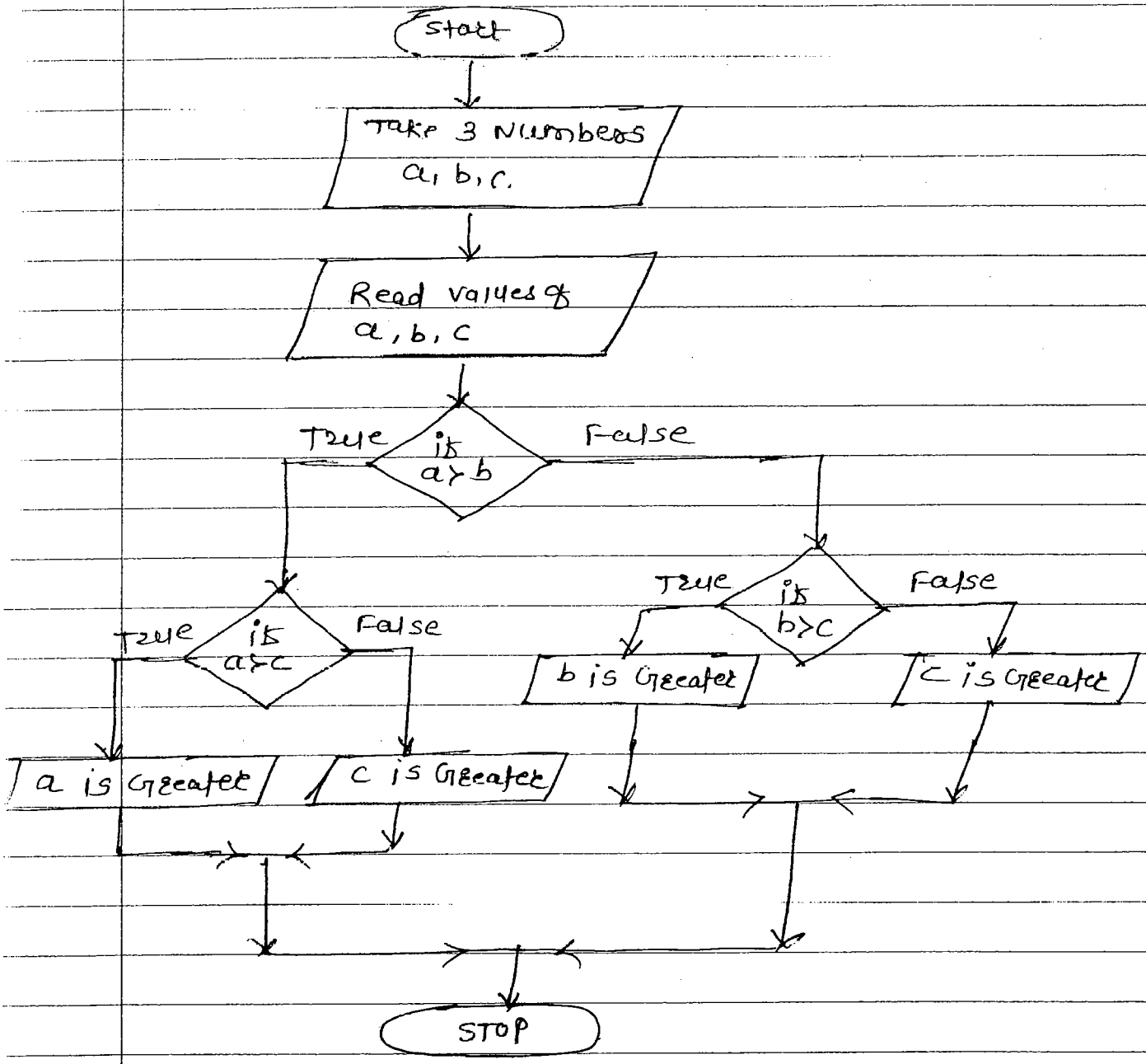
Flow chart.



* Program to find greater Number among three Numbers

```
#include<stdio.h>
void main()
{
int a,b,c;
Printf("\n Enter three Numbers\n");
scanf("%d%d%d",&a,&b,&c);
    if(a>b)
    {
        if(a>c)
        {
        Printf("\n a is greater Number\n");
        }
        else
        {
        Printf("\n c is greater Number\n");
        }
}
else
 {
        if(b>c)
        {
        Printf("\n b is greater number\n");
        }
        else
        {
        printf("\n c is greater Number\n");
        }
}
```

Flowchart


start
Take 3 Numbers
a, b, c.
Read values of
a, b, c
True
is
a>b
False
True
is
a>c
False
True
is
b>c
False
a is Greater
c is Greater
b is Greater
c is Greater
STOP

(2) ~~Using~~

- Another format of nesting of if-else

Syntax:-

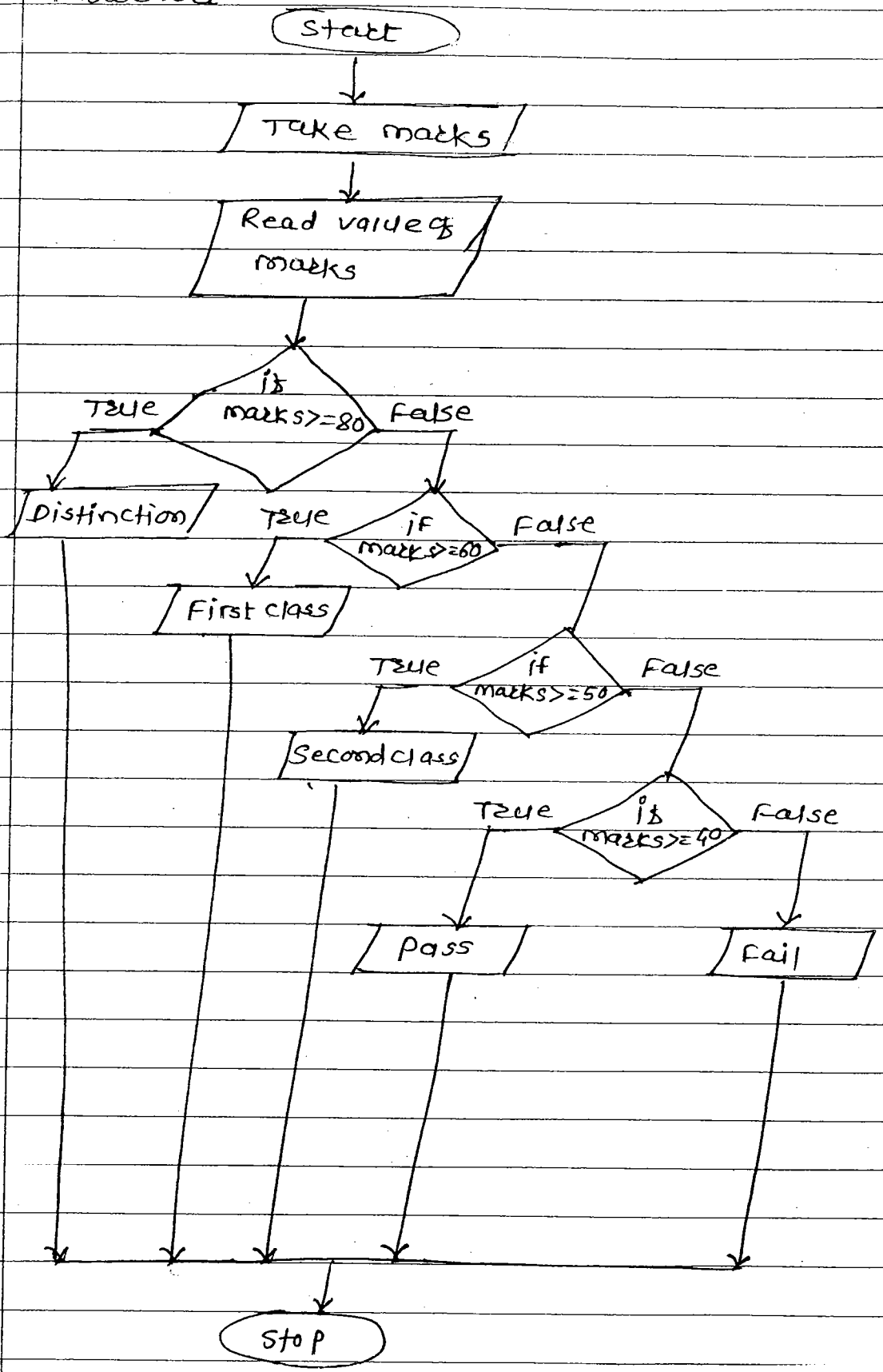
```
if (condition 1)
{
    ------
    ------
}
else
{
    if (condition 2)
    {
        ------
        ------
    }
    else
    {
        if (condition 3)
        {
            ------
            ------
        }
        else
        {
            if (condition 4)
            {
                ------
                ------
            }
            else
            {
                ------
                ------
            }
        }
    }
}
```

- Progeam to display grading system of students as per marks

```
#include <stdio.h>
void main()
{
int marks;
Printf("Enter your marks \n");
scanf("%d", &marks);
        if( marks >= 80)
        {
        Printf("\n Distinction \n");
        }
      else
        {       if(marks>=60)
                {
                Printf("\n First class\n");
                }
              else
                {       if (marks>=50)
                        {
                        Printf("\n second class\n");
                        }
                      else
                        {       if (marks>=40)
                                {
                                Printf("\n pass\n");
                                }
                              else
                                {
                                Printf("\n Fail\n");
                                }
                        }
```

Page No.

Flowchart



(2) Using else-if Ladder:-

When we want to write complete if-else inside the block of onther else then it is generally a else-if ladder. In else-if ladder processor first evaluate main condition if it is true then it executes instructions of its true block, if condition is false it jump to else i.e. false block of condition & there is again one more condition again processor evaluates next condition if it is true it executes true block that condition. otherwise if condition is false it again jumps to else & so on.
In else-if there is no use of curly bracket ({,}) when there is only on instruction in block of condition.
For using else-if in program we have to combine two keywords else & if together. i.e. else if.

Syntax :-
```
if (condition 1)
---------
else if (condition 2)
- - - - - - - - - -
else if (condition 3)
- - - - - - - - - -
else if (condition 4)
---------
else
- - - - - - - - - -
```

As shown above syntax there are four conditions, processor first evaluate condition 1 if it is true it executes true block if condition is false it jumps to next else if condition i.e. condition 2, if condition 2 is true it executes given statements otherwise it jumps to next else if & so on if all above conditions are false then it executes last else.

- Program to display grading system of students as per marks using ~~else~~ else-if ladder

```
#include<stdio.h>
void main()
{
int marks;
printf("\n Enter marks\n");
scanf("%d", &marks);
    if (marks >= 80)
    printf("\n Distinction\n");
    else if (marks >= 60)
    printf("\n First class\n");
    else if (marks >= 50)
    printf("\n second class\n");
    else if (marks >= 40)
    printf("\n pass\n");
    else
    printf("\n Fail");
}
```

- write a program to read the price of an item & display the discount as follows.

if price < 100 → 10% Discount
if price >= 100 && price < 500 → 20% discount
if price >= 500 && price < 1000 → 30% discount
if price >= 1000 → 40% discount.

```
void main()
{
int price;
printf("Enter price\n");
scanf("%d", &price);
    if (price < 100)
    printf("\n 10% Discount\n");
    else if (price >= 500 && price < 1000)
    printf("\n 30% Discount\n");
    else
    printf("\n 40% Discount\n");
}
```

* Looping Statements :-

In C following are the looping instructions

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

(1) while loop :-

while loop is a iterative type of instruction. To use while loop we are using while keyword. It is also called as top tested loop or entry controlled loop. When processor looks while keyword it immediately understand user want to perform repetition of some instructions. In this loop condition is given at the top.

Syntax :-

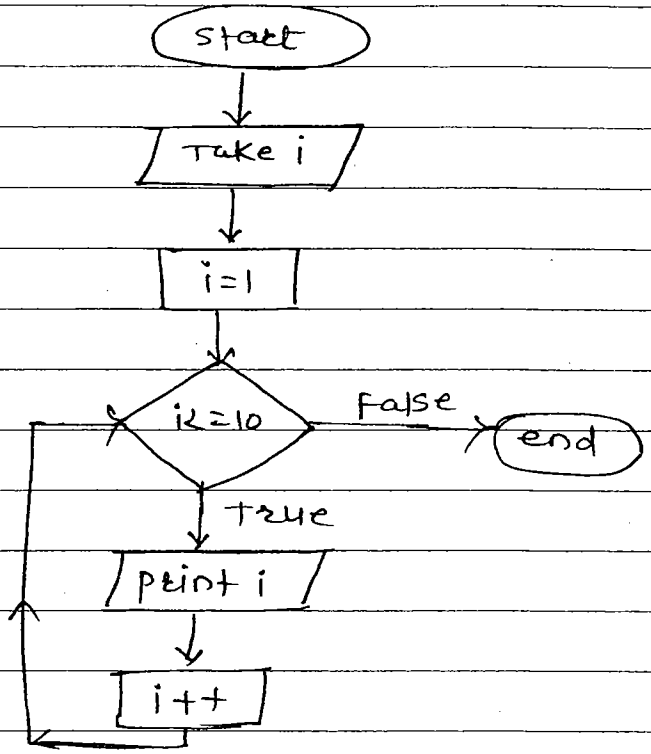
```
Initial No;
while (Condition)
{
    ---------  Block of while
    ---------
    ---------
}
Next statement;
```

As shown in syntax Condition is given at top & then there is block of instructions. Processor first evaluate condition, if condition is true then it execute given block of instructions repeatedly till condition becomes false. When condition is false it jumps to next statement.

Repetition of block of instruction is depend on condition.

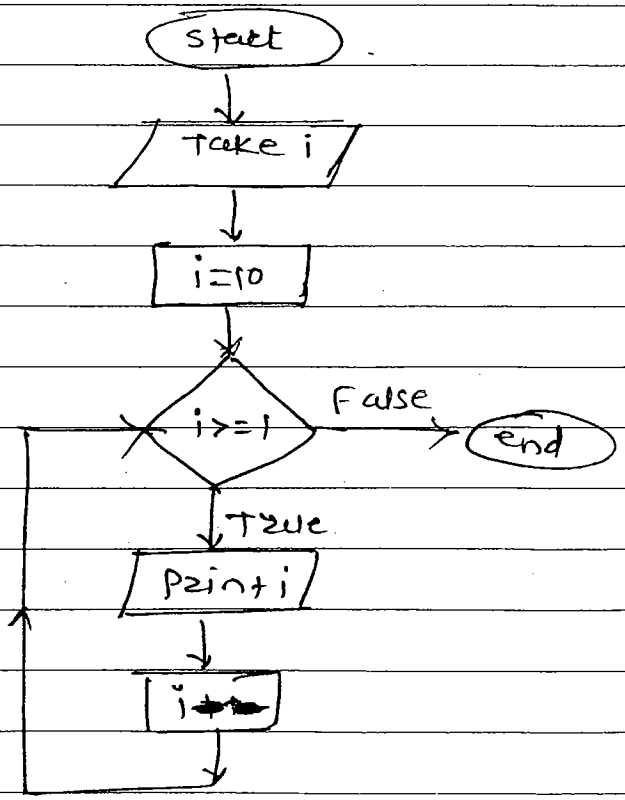
- Program to Display first 10 Natural numbers using while loop

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



- Program to display first 10 natural Numbers in Reverse order

```
#include<stdio.h>
void main()
{
int i;
i= 10;
while(i>=1)
{
 Printf("%d\n", i);
  i--;
}
}
```



(2) do while loop :-

This is also one of iterative statement. To use do while loop we are using do & while keywords. It also called as Bottom tested loop or exit controlled loop. In this loop condition is given at bottom.

Syntax :-
```
Initial number;
do
{
    ------  Block of do-while
    ------
}
while (Condition);
Next statement;
```

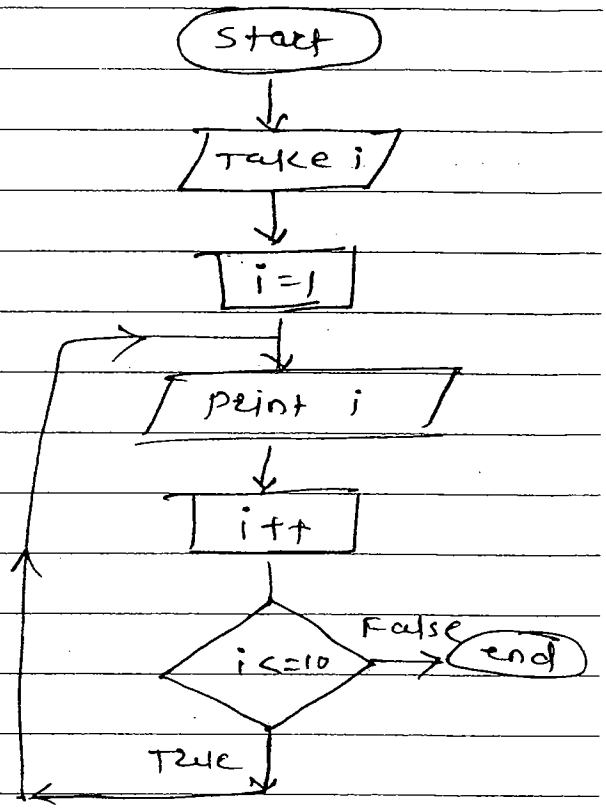
As shown in syntax condition is given at bottom & block of loop is given before condition with do keyword. processor first execute the block of do while then it evaluates the condition. If condition is true then it go back executes given block again & again it evaluates condition & so on.

Repetition of instructions is depend on condition is condition is true then processor performing iterations if condition is false then instead of going back the processor jumps to next statement after condition.

Note :- If initial condition is false then also atleast one iteration we are getting result with do..while loop.

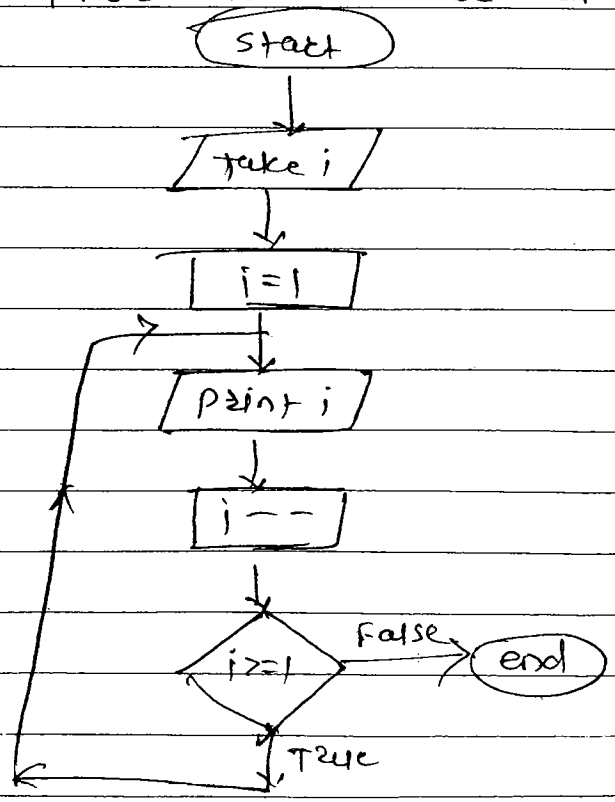
- Program to Display first 10 natural No's using do ..while

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



- Program to Display 10 natural No's ~~1 to~~ in reverse order

```
#include<stdio.h>
void main()
{
int i;
i=10;
do
 {
   Printf("%d\n", i);
   i--;
 }
 while(i>=1);
}
```



* Difference between while & do...while loop.

	while	do... while
(1)	In while loop Processor first evaluate condition, is condition is true then only it executes block of loop.	(1) In do...while loop Processor first execute the block of loop then evaluate condition. if condition is true, then only it perform further iterations.
(2)	It is entry controlled loop	(2) It is exit controlled loop.
(3)	This also called top tested loop	(3) This is called bottom tested loop.
(4)	In this only while keyword is used	(4) In this two keywords are used while & do.
(5)	There is no semicolon for condition	(5) There is semicolon for condition.
(6)	This should be used when condition is more important.	(6) This should be used when ~~condition~~ process is important
(7)	If initially condition is false, no any result we get	(7) If initially condition is false then atleast once we get result due to block of loop is before condition.

(8)	while (condition)
	{
	=
	=
	}
		(8) do
		{
		=
		=
		} while (condition);

(3) for loop

For loop is a most commonly used loop in C. It consist of three actions namely Initial no, condition & Increment Decrement. Each action is seperated by semicolon (;).

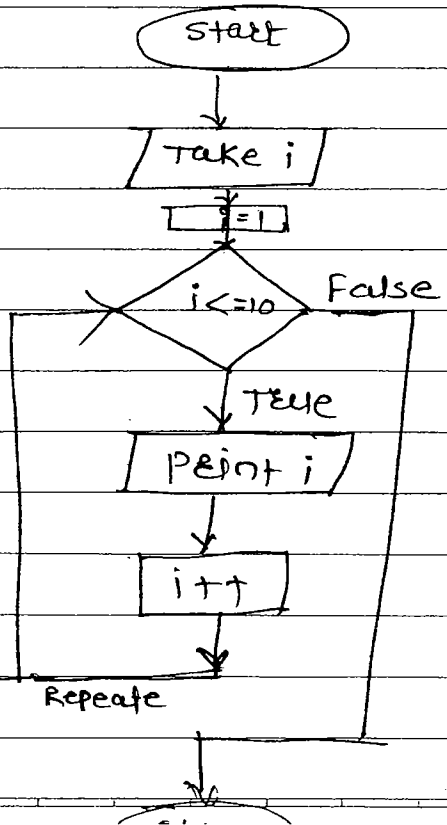
Syntax:—

```
for(Initial no ; Condition ; Increment / Decrement)
{
    - - - -
    - - - -        Block of for loop.
    - - - -
    - - - -
}
Next statement ;
```

As shown in syntax first processor evaluate initial no, then it checks the condition after evaluating condition if condition is true then it executes block of for loop, if condition is false it does not execute block of loop. After execution of block of loop then it go for increment/decrement when condition is false processor jumps to next statement.

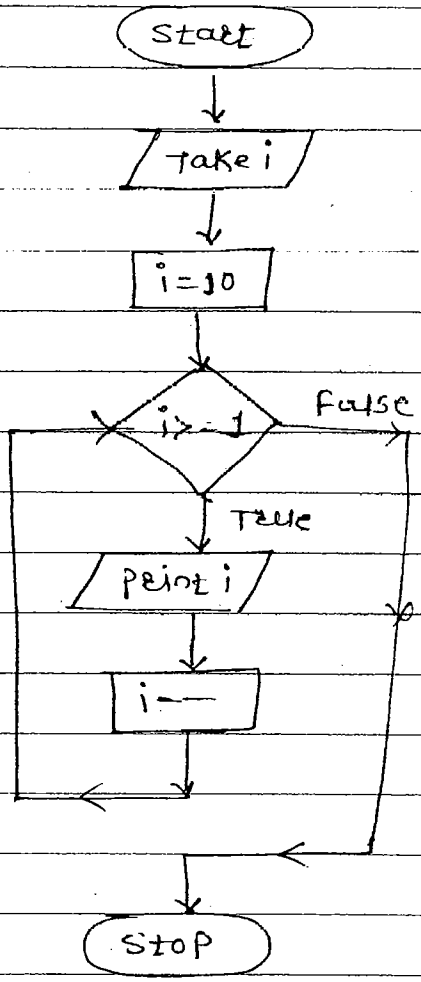
- Program to print first 10 natural no's using for loop

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



- Program to print first 10 natural no's in reverse order using for loop.

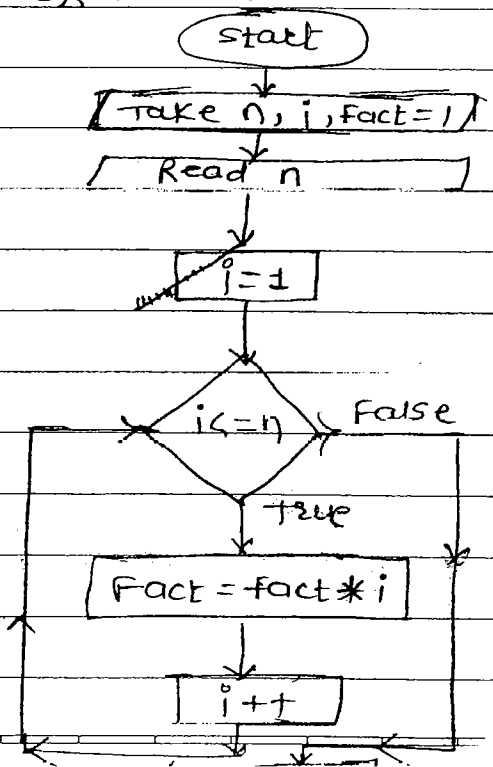
```
#include<stdio.h>
void main()
{
int i;
for(i=10; i>=1; i--)
{
  printf("%od", i);
}
}
```



* Some programs on Conditional & looping statements :-

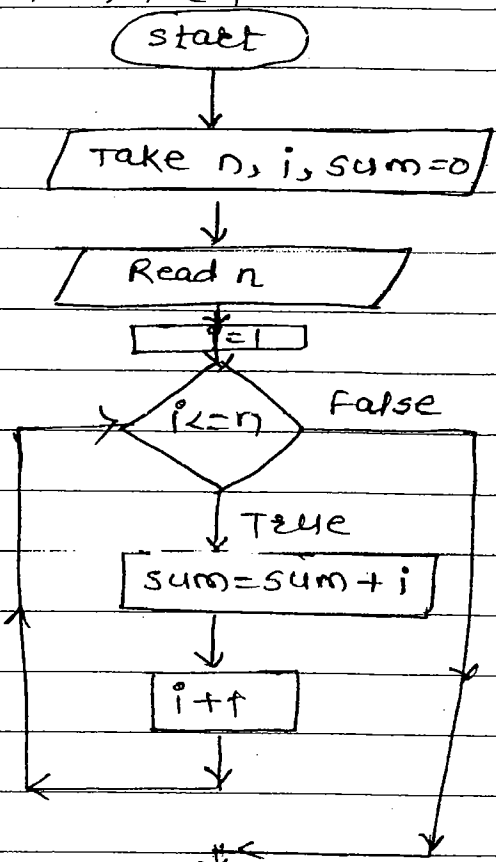
- Write program to print factorial of Number.

```
#include<stdio.h>
void main()
{
int n, i, fact=1;
printf("\n Enter Number\n");
scanf("%d", &n);
    for(i=1; i<=n; i++)
    {
     fact=fact * i;
    }
printf(" Factorial is %od", fact);
}
```



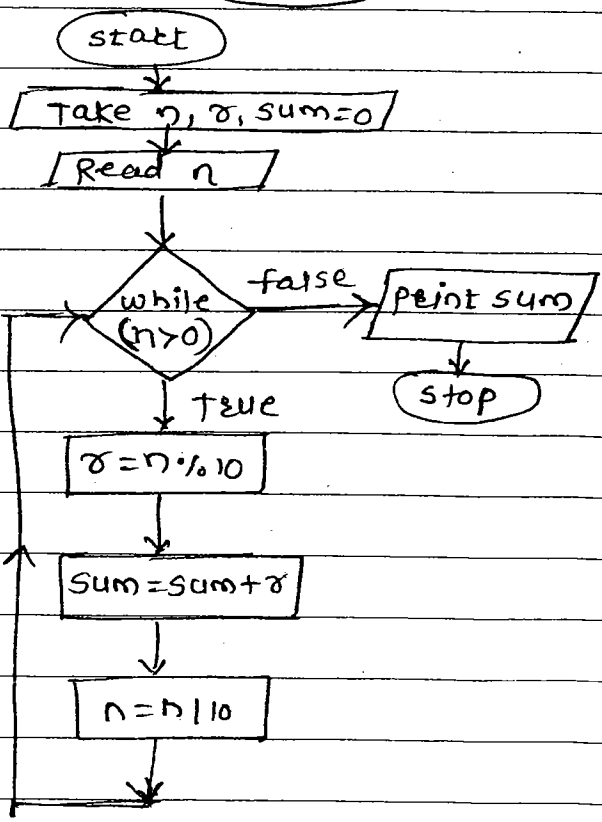
- Write a program to print sum of n natural no's

```
#include<stdio.h>
void main()
{
int n, i, sum=0;
printf("\n Enter Number\n");
scanf("%d", &n);
        for(i=1; i<=n; i++)
        {
            sum = sum + i;
        }
printf(" sum is %d", sum);
}
```



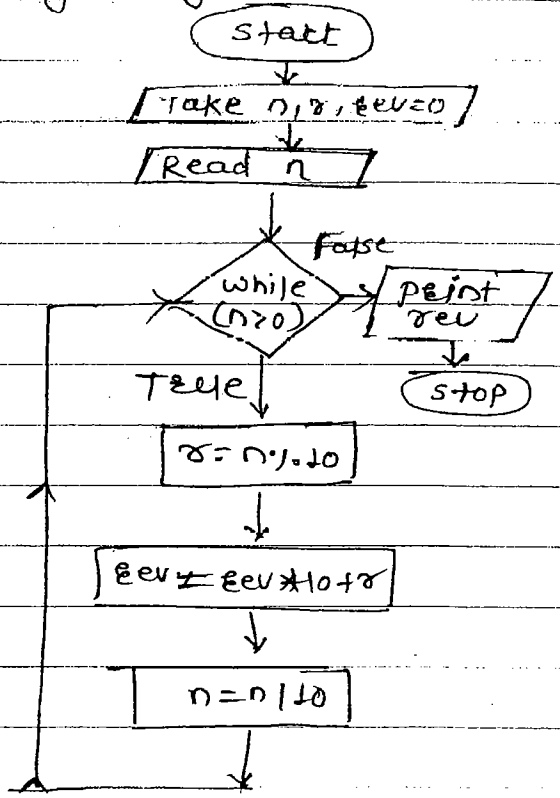
- If a 4-digit number is entered by user write a program to print sum of digits of entered number.

```
# include<stdio.h>
Void main()
{
int n, sum=0, r;
printf("\n Enter 4-digit number\n");
scanf("%d", &n);
        while(n>0)
        {
        r = n%10;
        sum = sum + r;
        n = n/10;
        }
printf("\n sum of digits of no is\n");
printf("%d \n", sum);
```



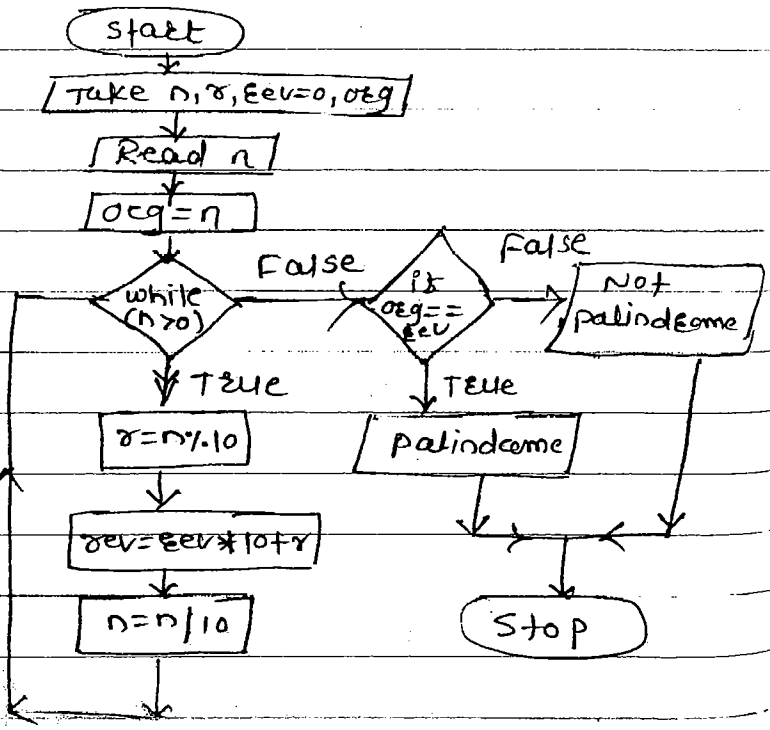
- If a four digit no is entered through keyboard. WAP to print reverse of that no.

```
#include<stdio.h>
void main()
{
int n, r, rev=0;
printf("\n Enter Number\n");
scanf("%d", &n);
        while(n>0)
        {
         r=n%10;
         rev = rev * 10 + r;
         n = n/10;
        }
 printf("\n Reverse of entered no is %d", rev);
}
```



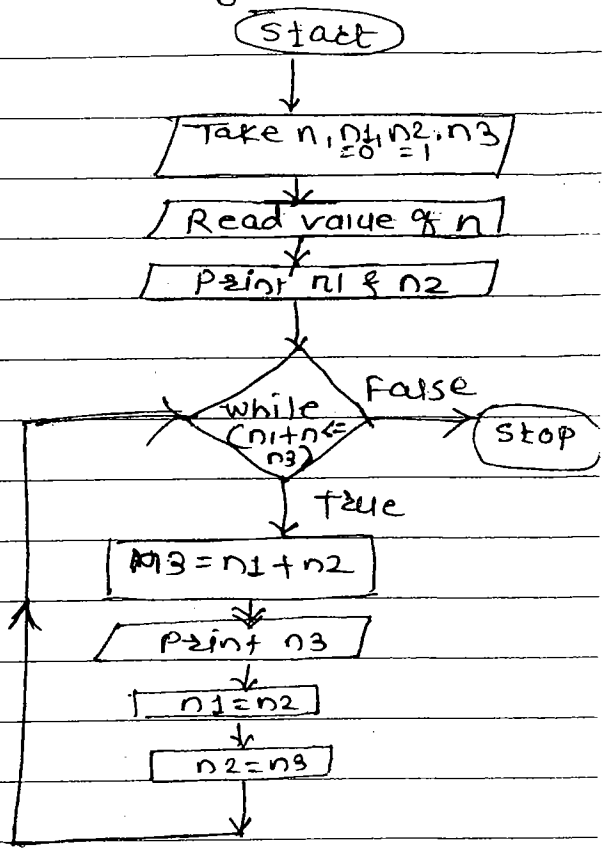
- Write a program to Display entered no is palindrome or not.

```
#include<stdio.h>
void main()
{
int n, r, rev=0, org;
printf("\n Enter number\n");
scanf("%d", &n);
org = n;
   while(n>0)
   {
 r=n%10;
rev = rev*10+r;
 n = n/10;
}
 if(org == rev)
  printf("Number is palindrome\n");
  else
  printf("Number is not palindrome\n");
```



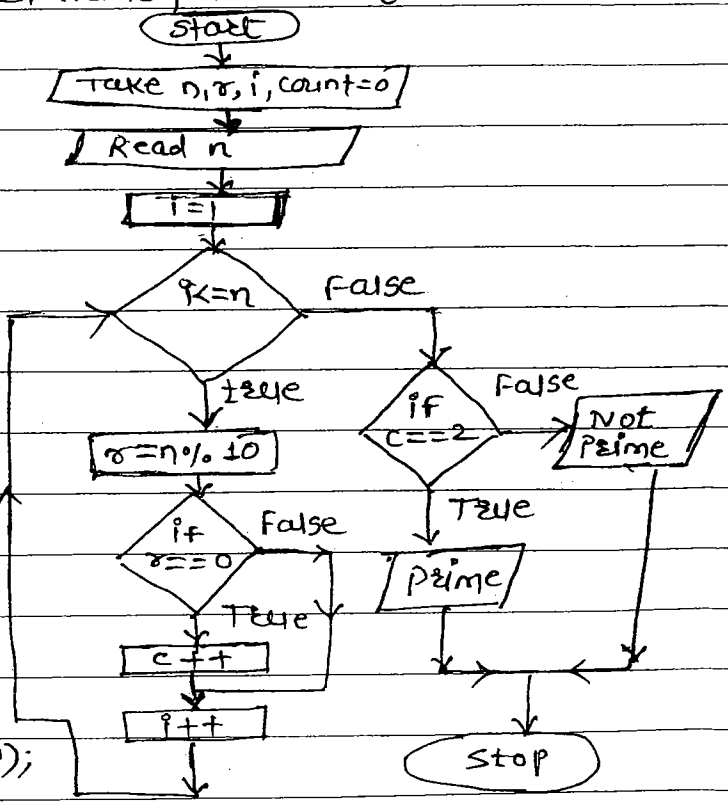
- write a program to display fibonacci series of number

```
#include<stdio.h>
void main()
{
int n, n1=0, n2=1, n3;
printf("\nEnter number\n");
scanf("%d", &n);
printf("%d %d", n1, n2);
    while(n1+n2<=n)
    {
    n3 = n1+n2;
    printf(" %d", n3);
    n1=n2;
    n2 = n3;
    }
}
```



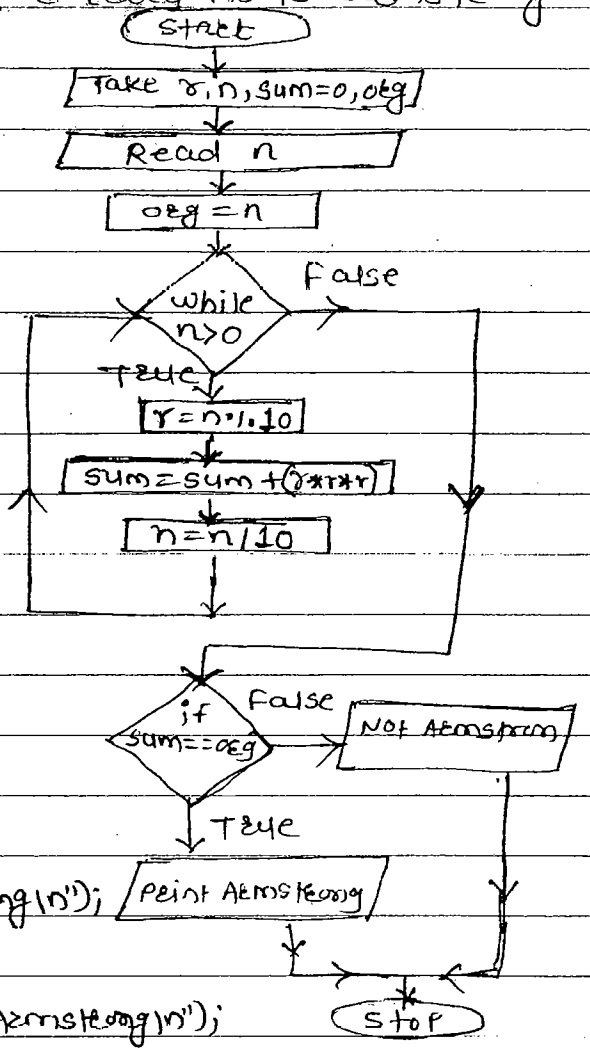
- write a program to check entered no is prime or not

```
#include<stdio.h>
void main()
{
int n, r, i, count=0;
printf("\nEnter Number\n");
scanf("%d", &n);
    for(i=1; i<=n; i++)
    {
        r= n%10;
            if(r==0)
            count++;
    }
if(count==2)
printf("\n Number is prime\n");
else
printf("\n Number is not prime\n");
```



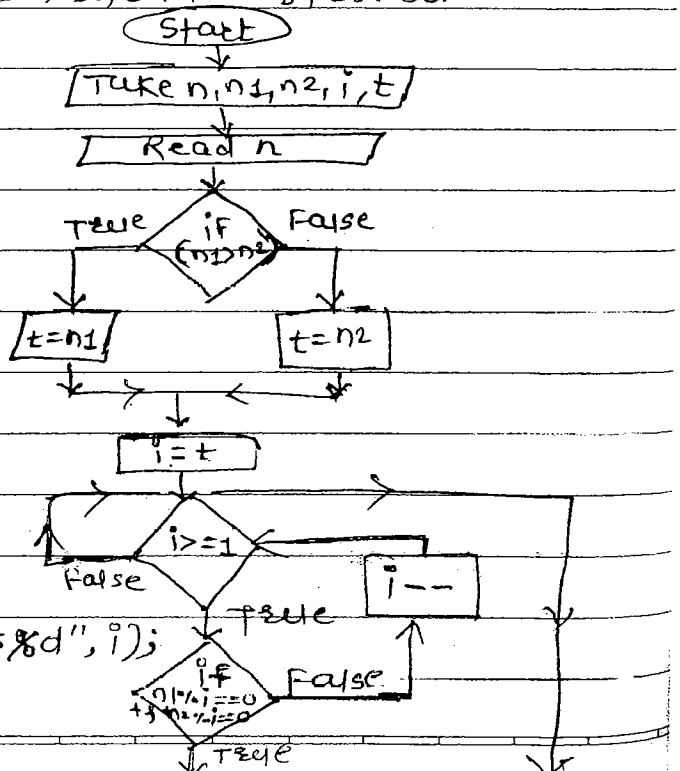
- write a program to check entered no is armstrong no or not

```
#include<stdio.h>
void main()
{
int r, n, sum=0, org;
printf("\n Enter Number\n");
scanf("%d", &n);
org = n;
        while(n>0)
        {
        r=n%10;
       sum= sum+(r*r*r);
       n=n/10;
       }
     if(sum==org)
     printf(" Number is Armstrong\n");
     else
     printf("\n Number is not Armstrong\n");
}
```



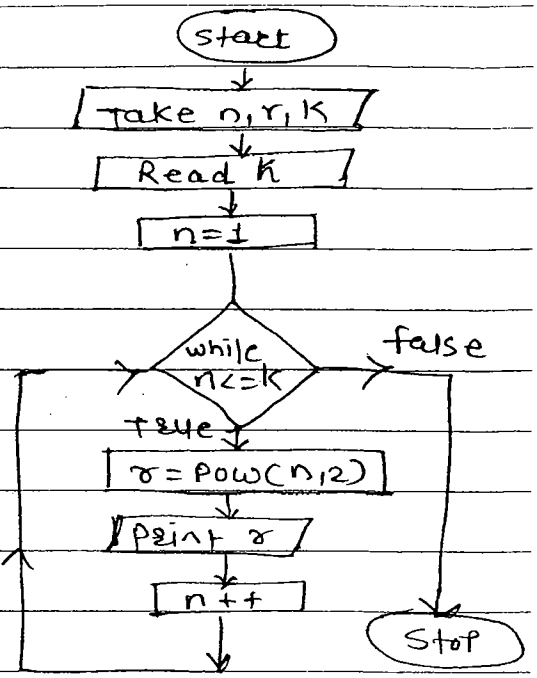
- write a program to print Greatest Common divisor of two No's.

```
#include<stdio.h>
Void main()
{
int n, n1, n2, i, t;
printf("\n Enter two Numbers\n");
scanf("%d%d", &n1, &n2);
t=(n1>n2) ? n1 : n2;
   for(i=t; i>=1; i--)
   {   if(n1%i==0 && n2%i==0)
       {
         printf("GCD of Numbers is %d", i);
         break;
       }
   }
```



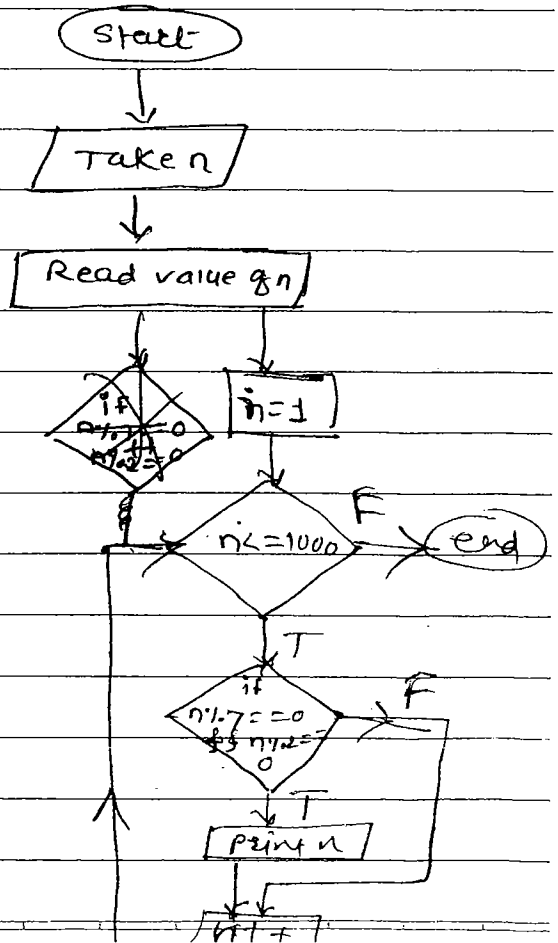
- write a program to print 1 4 9 16 25 ..... n.

```
#include<stdio.h>
#include<math.h>
void main()
{
int n, r, K;
Printf("\nEnter number\n");
Scanf("%d", &K);
n=1;
  while(n<=K)
  {
   r= pow(n,2);
   Printf("%d", r);
   n++;
  }
}
```



- write a program to display all even no's upto 1000 which are divisible by 7.

```
#include<stdio.h>
void main()
{
int n;
  for(n=1; n<=1000; n++)
  {
      if(n%7==0 && n%2==0)
      {
       Printf("%d ", n);
      }
}
```



* Nesting of for loop:-

Nesting of for loops indicates putting one for loop inside another for loop. We can display results in matrix format using nesting of for loop. When we are using only one for loop then we can display result either in row or in column but we can't display in matrix format like nesting of for loop.

Syntax:-

```
for(Initial No; condition; Incr/Decr)
{
    for(Initial No; condition; Incr/Decr)
    {
        ---------
        ---------
        ---------
    }
}
```

Outer Loop

Inner Loop

Block of loop

As shown in syntax there are 2 for loops inner & outer loop. First processor meet to outer for loop, it initialize the no, then it checks the outer condition, is condition is true then it comes to meet to inner loop, it initialize the no again it checks the inner condition, is inner condition is true it execute Block of loop then it do Incr/Decr then by taking incremented value it iterates inner loop repeatedly until condition of inner loop will be false. When inner condition is false then it come out of inner loop & it do incr/Decr of outer loop & by taking incremented or Decremented value it iterate the outer loop, this process repeat again & again till outer condition goes to false.

①
```
#include<stdio.h>
void main()
{
int i,j;
for(i=1; i<=5;i++)
{
    for(j=1; j<=5; j++)
    {
        printf(" *");
    }
    printf("\n");
}
```

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

* FLOYDS TRIANGLE

②
```
#include<stdio.h>
void main()
{
int i,j,n;
printf("\n Enter No.of rows\n");
scanf("%d",&n);
  for(i=1; i<=n; i++)
  {
      for(j=1; j<=i; j++)
      {
        printf(" *");
      }
    printf("\n");
  }
}
```

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

(3)

```
#include<stdio.h>
void main()
{
int i,j;
 for(i=5; i>=1 ; i--)
 {
     for(j=1;j<=i; j++)
     {
       printf(" *");
     }
 printf("\n");
}
}
```

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

(4)

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

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

```
#include <stdio.h>
void main()
{
int i,j, n=15;
for(i=5; i>=1 ; i--)
{
    for(j=1; j<=i; j++)
    {
      printf("%d",n);
      n--;
    }
 printf("\n");
}
}
```

```
15 14 13 12 11
10 9 8 7
6 5 4
3 2
1
```

```
#include <stdio.h>
void main()
{
int i,j,n;
printf("\n Enter no. of rows\n");
scanf("%d",&n);
for(i=1; i<=n; i++)
{
    for(j=1; j<=i ;j++)
    {
     printf("%d", i);
    }
 printf("\n");
}
}
```

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

```
#include<stdio.h>
void main()
{
int i,j;
for(i=5; i>=1 ;i--)
 {
    for(j=1; j<=i;j++)
    {
     printf("%d",i);
    }
printf("\n");
}
}
```

```
5 5 5 5 5
4 4 4 4
3 3 3
2 2
1
```

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

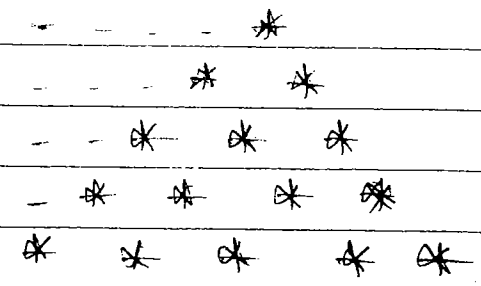
```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

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

```
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

Pyramid pattern

```
#include <stdio.h>
void main()
{
int n, space, i, j, s;
printf("\n Enter no. of rows\n");
scanf("%d", &n);
space = n-1;
 for(i=1; i<=n; i++)
  {
        for(s=1; s<=space; s++)
         {
           printf(" ");
         }
        for(j=1; j<=i; j++)
         {
           printf(" *");
         }
    printf("\n");
    space = space - 1;
  }
}
```



Pascal Triangle Pattern

```
#include <stdio.h>
void main()
{
int n, space, i, j, s, a;
Printf("\n Enter no. of rows\n");
scanf("%d", &n);
space = n-1;
  for(i=0; i<n; i++)
  {
      for(s=0; s<space; s++)
      {
          printf("  ");
      }
      a=1;
      for(j=0; j<=i; j++)
      {
          printf("%d", a);
          a=(a*(i-j)/(j+1));
      }
  printf("\n");
  space--;
  }
}
```

```
    1
   1 1
  1 2 1
 1 3 3 1
1 4 6 4 1
```

* Switch case statement :-

The switch case statement is used when we want to select a choice from no. of choices (cases). This statement selects a single choice (case) from set of available alternatives.

The switch case is also a control flow statement but using switch case we can't check condition & the in switch case a single expression matches with cases. If it mathes with any one of the case then corresponding block of statements is executed & processor breaks the switch case.

Syntax :-

```
switch (expression)
{
case 1: ---
        break;
case 2: ---
        break;
case 3: ---
        break;
  :
case n: ---
        break;
default: ---
        break;
}
```

As shown in above syntax switch is keyword to indicate switch case statement, case is a keyword to displayes no. of cases, break keyword is used to break respective case, & default keyword is used to execute default case.

TO take a choice from user we have to declare variable used in swith case either integer or character because case number it may be 1,2,3,4.... or it may be a,b,c,d or A, B, C, D.

The switch case statement taking choice from user if it matches the value against the case number, if a match is found betn expression of one of case, then corresponding case is executed & break statement break respective case. If match not found with any case then default statement get execute.

Example

```
#include<stdio.h>
void main()
{
int choice;
printf("\n 1. for APPLE \n 2. For mango \n 3. For Orange \n\n");
printf("\n Enter your choice \n");
scanf("%d", &choice);
        switch(choice)
        {
      case 1: printf("\n I Like APPLE \n");
              break;
      case 2: printf("\n I Like mango \n");
              break;
      case 3: printf("\n I Like Orange \n");
              break;
      default: printf("\n Wrong Choice \n");
              break;
    }
}
```

Page No.:

- Write a menu driven program having following menus
  ① Addition ② substraction ③ Exit.

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
char choice;
int a, b, Add, sub;
printf("A. For Addition \n B. For substraction \n C For Exit\n");
printf("\n Enter Your choice\n");
scanf("%d", &choice);
    switch( choice)
    {
  Case 'A': printf("\n Enter two Numbers\n");
            scanf("%d%d", &a, &b);
            Add = a+b;
            printf("\n Addition is %d\n", Add);
            break;
  case 'B': printf("\n Enter two Numbers\n");
            scanf("%d%d", &a, &b);
            sub = a-b;
            Printf("\n substraction is %d\n", sub);
            break;
  Case 'C': exit(0);
            break;
  default: printf("\n Wrong Choice\n");
           break;
}
}
```

- Write a menu driven program having following options:
  1. Factorial of Number
  2. even or odd
  3. prime or not.

```
#include<stdio.h>
void main()
{
int choice, n, i, fact=1, r, count=0
printf("1. For Factorial \n 2. For Even or odd \n 3. For
        Prime or not \n \n");
printf("\n Enter your choice \n");
scanf("%d", &choice);
    switch(choice)
    {
    case 1: printf("\n Enter Number \n");
            scanf("%d", &n);
            for(i=1; i<=n; i++)
            {
             fact = fact*i;
            }
            printf("Factorial is %d\n", fact);
            break;
    case 2: printf("\n Enter the Number \n");
            scanf("%d", &n);
             if(n%2==0)
             {
             printf("\n Even \n");
             }
             else
             {
             printf("\n Odd \n");
             }
             break;
```

```
case 3: printf("\n Enter number\n");
        scanf("%d", &n);
        for(i=1; i<=n; i++)
        {
            r = n%i;
            if(r==0)
            {
                count++;
            }
        }
        if (count==2)
        {
            printf("\n Number is prime\n");
        }
        else
        {
            printf("\n Number is not prime\n");
        }
        break;
default: printf("\n Wrong choice\n");
        break;
    }
}
```

- Write a menu driven program having following menus
① Area of circle ② Area of Triangle ③ Area of Rectangle
④ perimeter of circle ⑤ volum of cube.

Break & Continue statement

Explain Difference between break & Continue statement

	Break	Continue
1.	This statement exit from loop or switch case when it gets executed.	1. This statement goes back to the loop without breaking loop.
2.	When processor reach to break statement while execution it stops loop from current iteration.	2. When processor reach to Continue statement while execution it stops only current iteration & it goes back to perform further iteration
3.	keyword used break.	3. keyword used continue.
4.	This statement can be used either inside loop or inside switch case	4. This statement can be used only inside loop, when there is requirement.
5.	syntax of break break;	5. Syntax of Continue continue;

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

* goto statement :-

The goto statement transfer the control from one statement to another in same program. By using goto statement & corresponding label the processor jumps from one position to another position in program. TO use goto statement we can use goto keyword. label is a user defined entity where processor is transferring control.

- goto syntax :-

```
goto label;

label: ...
```

As shown in above syntax goto is a keyword to use goto statement. When processor reach to goto while execution it jumps to respective label & associated statements with that label gets execute.

* There are two types of goto statement
(1) Conditional goto
(2) unconditional goto.

(1) unconditional goto :-

When goto statement is used without any condition then it is called as unconditional goto. When such type goto statement gets executed then control transfer directly to respective label without checking condition.

ex:-
```
#include<stdio.h>
void main()
{
printf("\n Hi\n");
goto Bottom;
Bottom : printf("\n Hello\n");
}
```

(2) Conditional goto

When goto statement is used with condition then it is called as conditional goto statement. In this type processor checks condition, if condition is true then goto statement is executed.

ex:-
```
#include<stdlib.h>
#include<stdio.h>
void main()
{
int n;
printf("\n Enter Number\n");
scanf("%d", &n);
if (n%2==0)
{
goto even;
}
else
{
goto odd;
}
even : printf("\n Even Number\n");
goto end;
odd : printf("\n odd Number\n");
goto end;
```