UNIT - 3

Control Flow

☆ Introduction

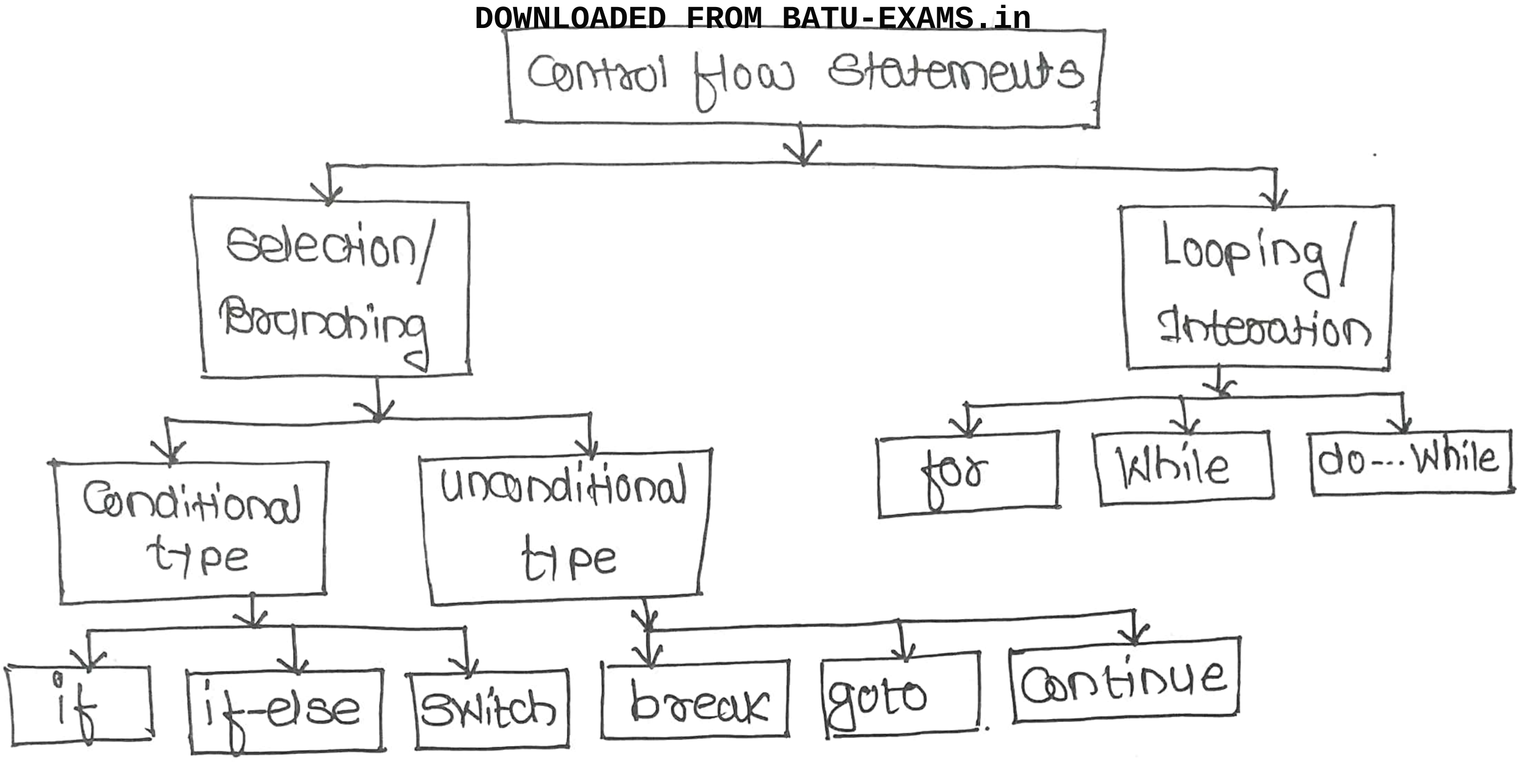
- A statement is the smallest executable unit of a C language program and it is terminated with a semicolon (;).

- Statement is an instruction given to the computer to perform a particular task like reading input, displaying output or evaluating an expression etc.

- By default, statements in a C program are executed in a sequential order. The order in which the program statements are executed is known as 'flow of program control' or just 'Flow of control'. By default, the program control flows sequentially from top to bottom.

- All programs that we have developed till now have default flow of control. Many practical situations like decision making, repetative execution of a certain task, etc. require deviation or alteration from the default flow of program control.

- The default flow of control can be altered by using flow control statements.



☆ Simple Statements :-

— The simplest kind of statement in C is an expression (followed by a semicolon, the terminator for all simple statements). It's value is computed and discarded.

Example:

```
X=2;            /* an assignment statement */
 X = 2+3;      /* another assignment statement */
 2+3;
/* has no effect --- Will be discarded by smart compilers */
 Put ("hi");    /* a statement containing a function call */
 root2= sqrt(2);
 /* an assignment statement with a function call */
```

— Most statements in a typical C program are simple statements of this form.

- Other examples of simple statements are the jump statements return, break, continue, and goto.
- A return statement specifies the return value for a function (if there is one), and when executed it causes the function to exit immediately.
- The break and continue statements jump immediately to the end of a loop or the next iteration of a loop; We'll see about these detail when we study about loops. The goto statement jumps to another location in the same function, and exits for the rare occasions when it is needed. Using it in most circumstances is not suggestive.
- Statements in a program normally executed sequentially in the order in which they appear in the program. But number of times it is necessary to change the sequence of program, or to repeat some statements number of times.
- C language allows the user for decision making and change the sequence of the program using control or decision making statements.

☆ Control Statements

Use of control statements

- Control statements are used to control the sequence of statements' execution.

Classification of control statements

— These statements are broadly divided into four categories: Decision making statements, Selection statements, Looping statements, and Jump statements. as shown in following figure.

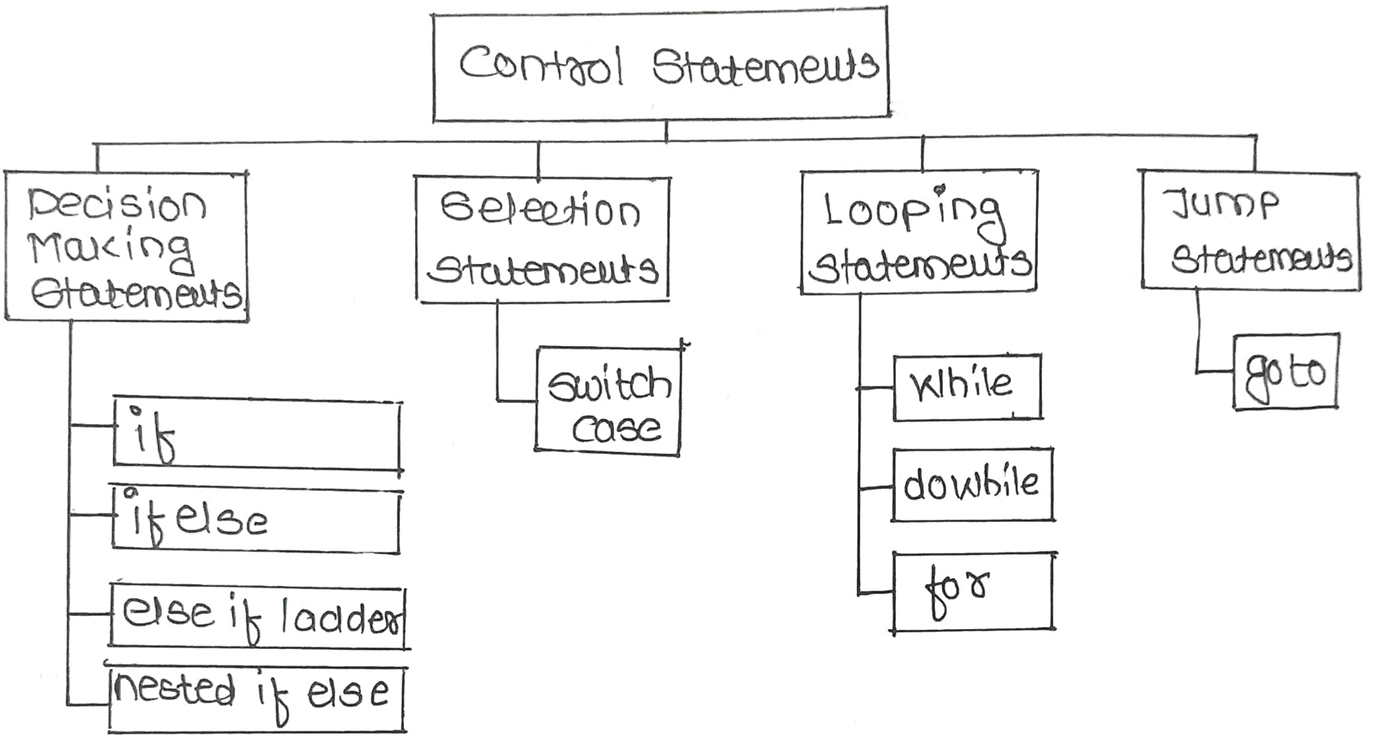


figure:- Various Control Statements

* Decision Making Statements

Use

The decision making statements are used to check different given conditions depending upon which the flow of control can be decided.

Four Decision Making statements

There are four decision making statements: if, if else, else if ladder, and nested if else.

✲ if Statement :-

Use of if statement -

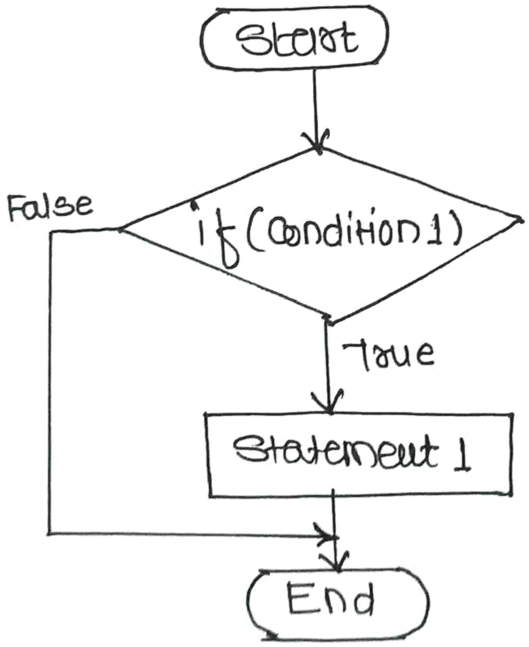
"if" is a decision making statement. It is generally used when we want to check single condition. If it is satisfied given code gets executed.

Sintax of if statement -

general form of following decision-making statement - if.

```
if (condition 1)
{
    Statement 1;
}
```

flowchart of if statement -



⇨ If condition 1 satisfies then statement 1 is Executed

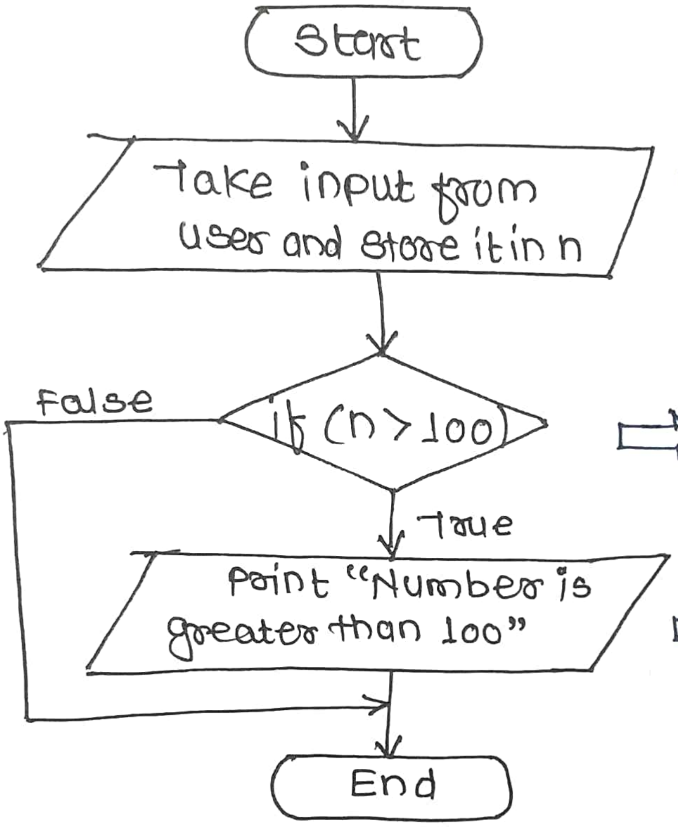
⇨ If condition 1 is not satisfied then it will skip statement 1

fig:- Flowchart of if statement.

Program:- Write a program to accept a number from user and check whether it is more than 100.

Solution ⇨

Flowchart:-


Start
Take input from user and store it in n
False
if (n>100)
True
Print "Number is greater than 100"
End

⇨ Accepts a number from user and store it in variable n

⇨ if n>100 condition satisfy it will execute statements in if block

⇨ If n>100 condition is not satisfy it will skip the if block statements

Program -

```
#include <stdio.h>
#include <conio.h>
int main ()
{
    int n;
    printf("\n Enter a number:");
    scanf("%d", &n);
    if (n>100)
    {
        printf("\n Number is greater than 100");
    }
    return 0;
}
```

Output -

```
Enter a number: 120
Number is greater than 100.
```

- Now in this example user enters 120 which is greater than 100, so the program print message.
- When the user enters value less than 100 then nothing is executed.
- This happens because we have given statement to execute if the condition satisfies, but if the condition does not satisfy then what to do? that we do not tell the compiler. This can be done using "else" statement.

Explanation -

- Here program accepts a number from user.
- If the number is greater than 100, it will print the message - "Number is greater than 100."
- If number is less than 100, message will not be printed.

✯ if-else Statement :-

Use

- if-else statement is used to tell the compiler What to do in both situations:
  (1) If the given condition satisfy
  (2) And if the given condition does not satisfy.
- If the given condition is true, then the "if" block of code will be executed, otherwise "else" block of code will be executed.

- In C programming language any non-zero and non-null value is considered as true, and for false either zero or null value is considered.

Syntax of if-else statement

```
if (condition 1)
{
    statement 1;
}
else
 {
   Statement 2;
 }
```

Flowchart of if else statement

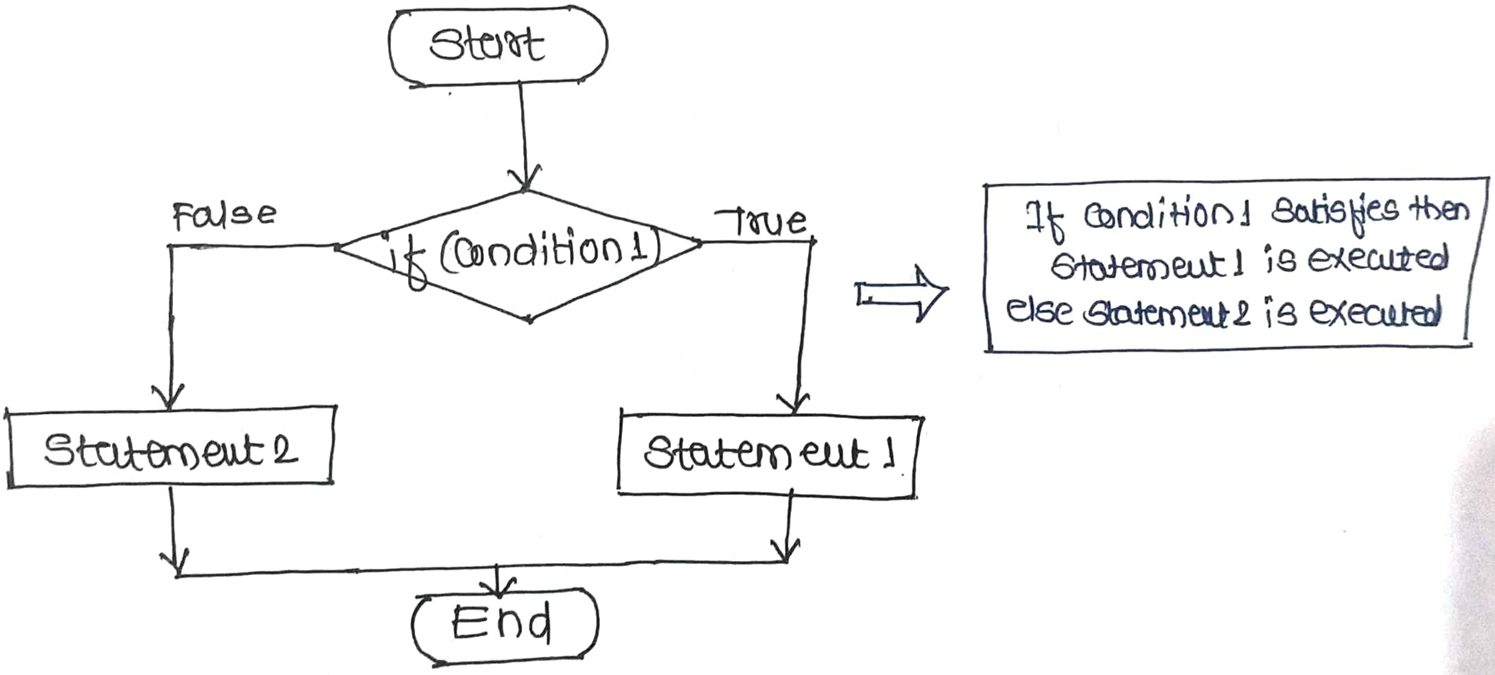


figure:- Flowchart of if-else statement.

Program - Write a program to accept a number from user & check whether it is more than 100. If the given is less than 100 then print another message.

or

give the example (program) of if-else statement.

Sol^{n} ⇒

Flowchart -

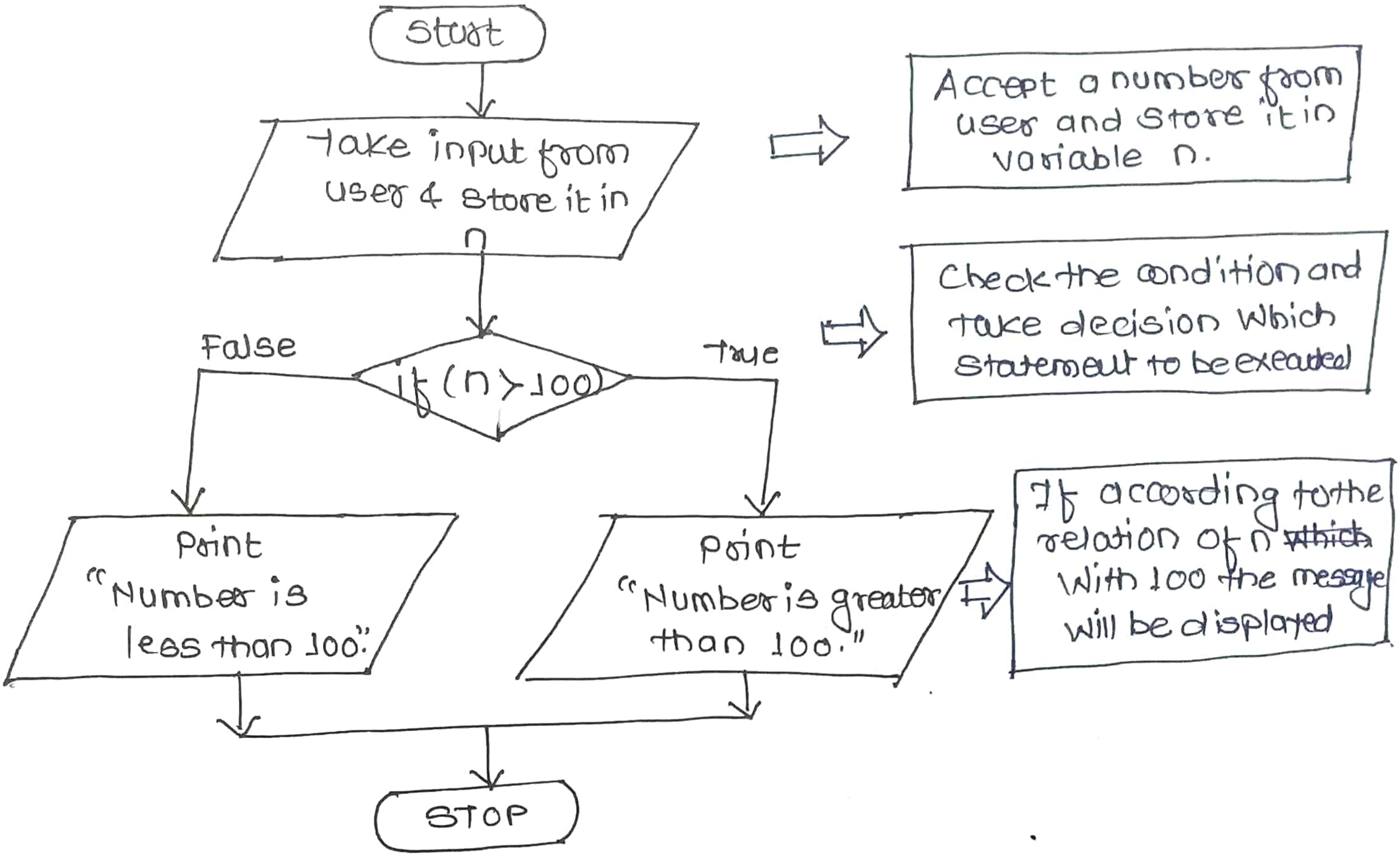

Start
Take input from user & store it in n
Accept a number from user and store it in variable n.
False
if (n>100)
True
Check the condition and take decision which statement to be executed
Print "Number is less than 100."
Print "Number is greater than 100."
If according to the relation of n ~~which~~ with 100 the message will be displayed
STOP

fig:- Flow chart of program.

Program:-

```
#include <stdio.h>
#include <conio.h>
main()
 {
  int n;
  printf("\n Enter a number: ");
  scanf("%d", &n);
  if (n>100)
  {
   printf("\n Number is greater than 100.");
  }
```

Includes header files

Accept number from user

If the condition is true this message will be displayed

```
else
 {
   printf("\n Number is less than 100.");
 }
}
```

If condition is false then this message will be displayed

Output:-

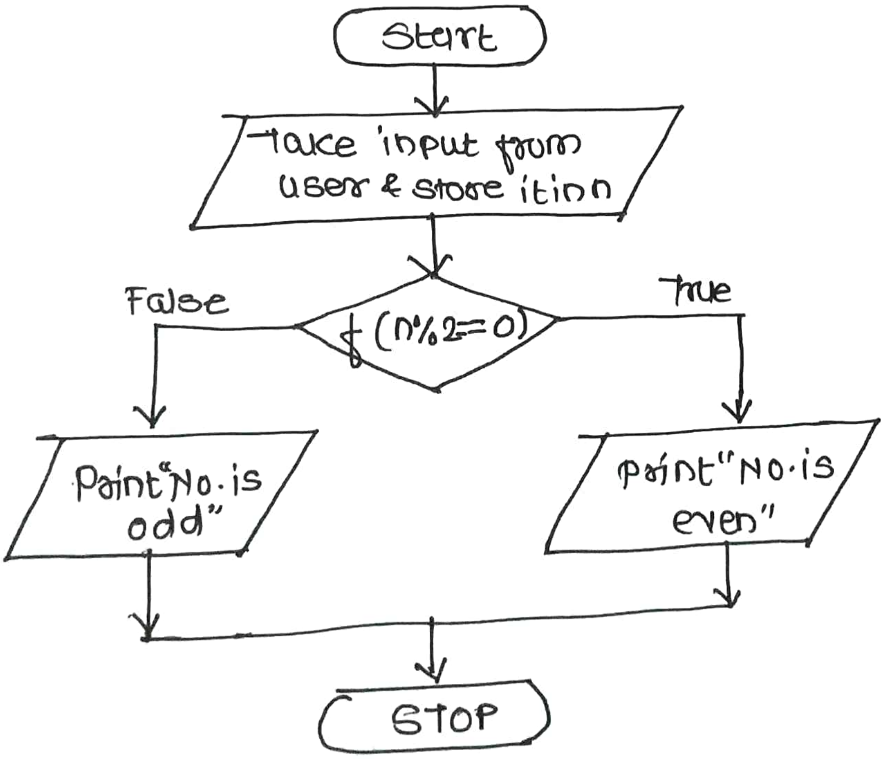
```
Enter a number : 50
Number is less than 100
```

Explanation -

- Here program accepts a number from user.
- If the number is greater than 100, it will print the message - " Number is greater than 100."
- If number is less than 100, it will print the message - " Number is less than 100."

———————— xox ————————

Program:- Write a program to accept a number from user and check whether it is even or odd.

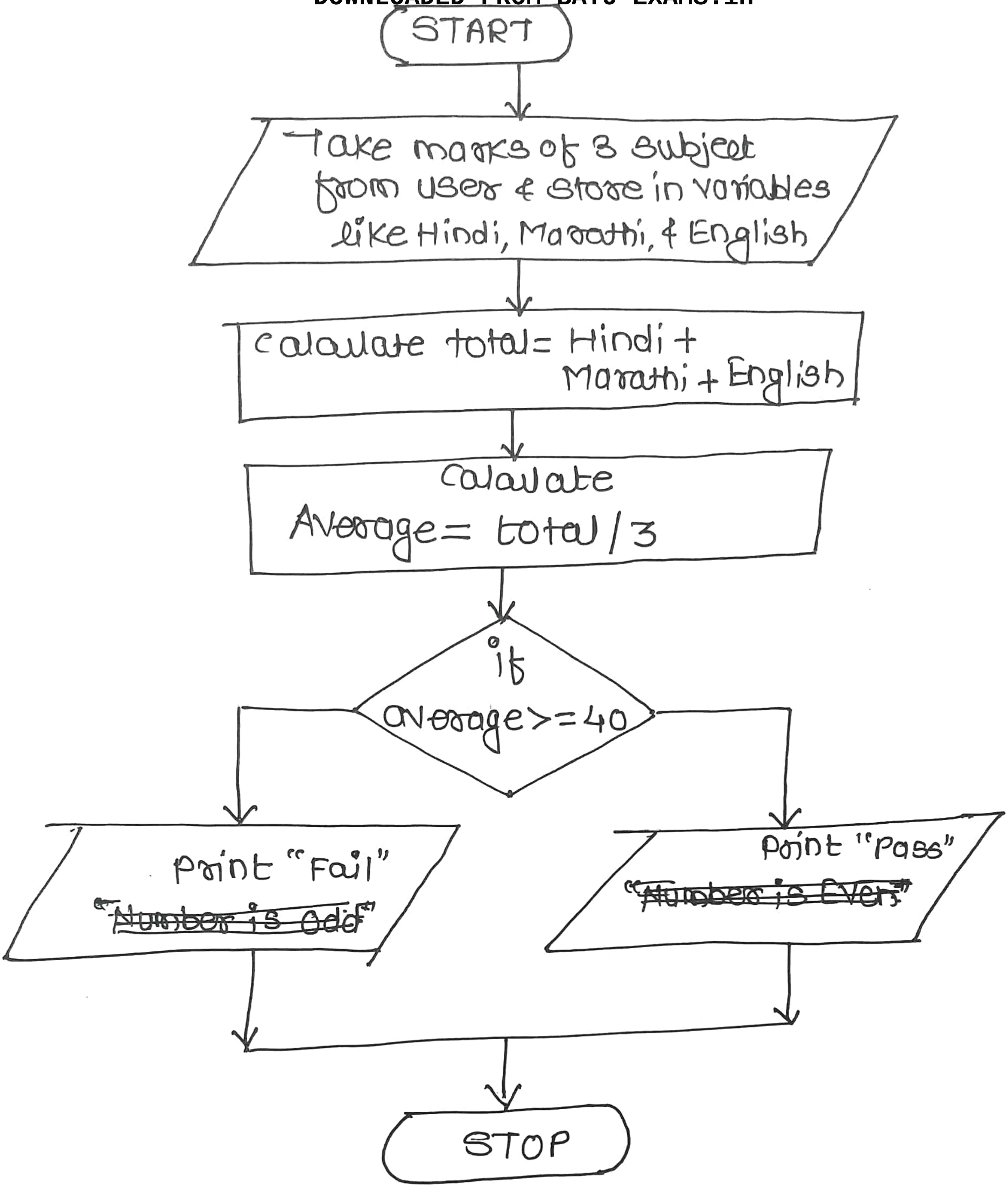


```
#include <stdio.h>
#include <conio.h>
 main()
 {
  int n;
  printf("\n Enter a Number: ");
  scanf("%d", &n);
  if (n%2==0)
   {
    printf("\n Number is even.");
   }
  else
    {
     printf("\n Number is odd.");
    }
 }
```

Output:-

```
 Enter a Number : 4
  Number is even.
```

Program:- Write a program to accept marks of 3 subjects from student. calculate the total and average of marks. If the average is >= 40 then give the remark as pass otherwise fail.


START
Take marks of 3 subject from user & store in variables like Hindi, Marathi, & English
calculate total= Hindi + Marathi + English
Calculate Average= total/3
if average>=40
print "Fail"
~~"Number is Odd"~~
Print "Pass"
~~"Number is Even"~~
STOP

Program:-

```
#include<stdio.h>
#include<conio.h>
 main()
  {
   int hindi, marathi, english, total, average;
   ~~Printf("\n Enter a number:");~~
   ~~scanf("%d", &n);~~
Printf("\n Enter a marks of 3 subjects : ");
Scanf("%d %d %d", &hindi, &marathi, &english);

total = hindi + marathi + english;
 average = total/3;

  Printf("\n Total marks : %d", total);
  Printf("\n Average : %d", average);

  if (average > 40)
   {
     Printf("\n Pass");
    }
  else
   {
    Printf("\n Fail");
    }
  }
```

Output :-

```
Enter a marks of 3 subjects : 90 80 70
Total marks : 240
 Average : 80
 Pass.
```

* else-if Ladder Statement :-

Use :-

- The "else if" ladder is used to test set of conditions in a sequence. It is also considered as multi-way decision making statement.

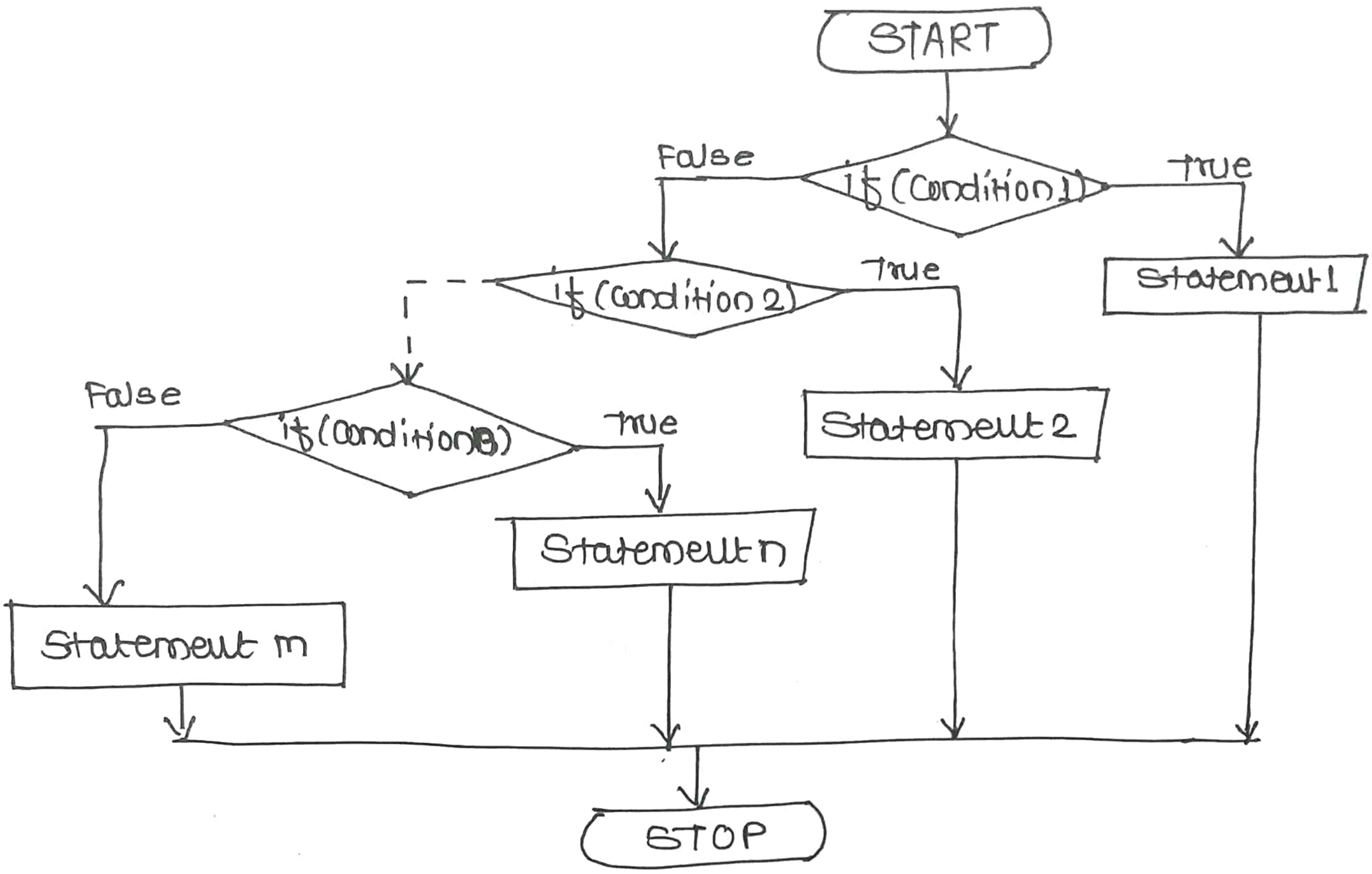
- Number of conditions are given in a sequence with subsequent statements.
- If any of the given condition is satisfied then the related statements are executed and the control exists from the else if ladder. That means further conditions are not going to be checked. But if the condition does not satisfy then the compiler goes to next condition to check the condition.

Syntax of else if ladder :-

```
if (condition 1)
  {
   Statement 1;
  }
else if (condition 2)
  {
   Statement 2;
  }

  - - - - - - - - - - -
  else if (condition n)
   {
   Statement n;
   }
  else
   {
   Statement m;
   }
```

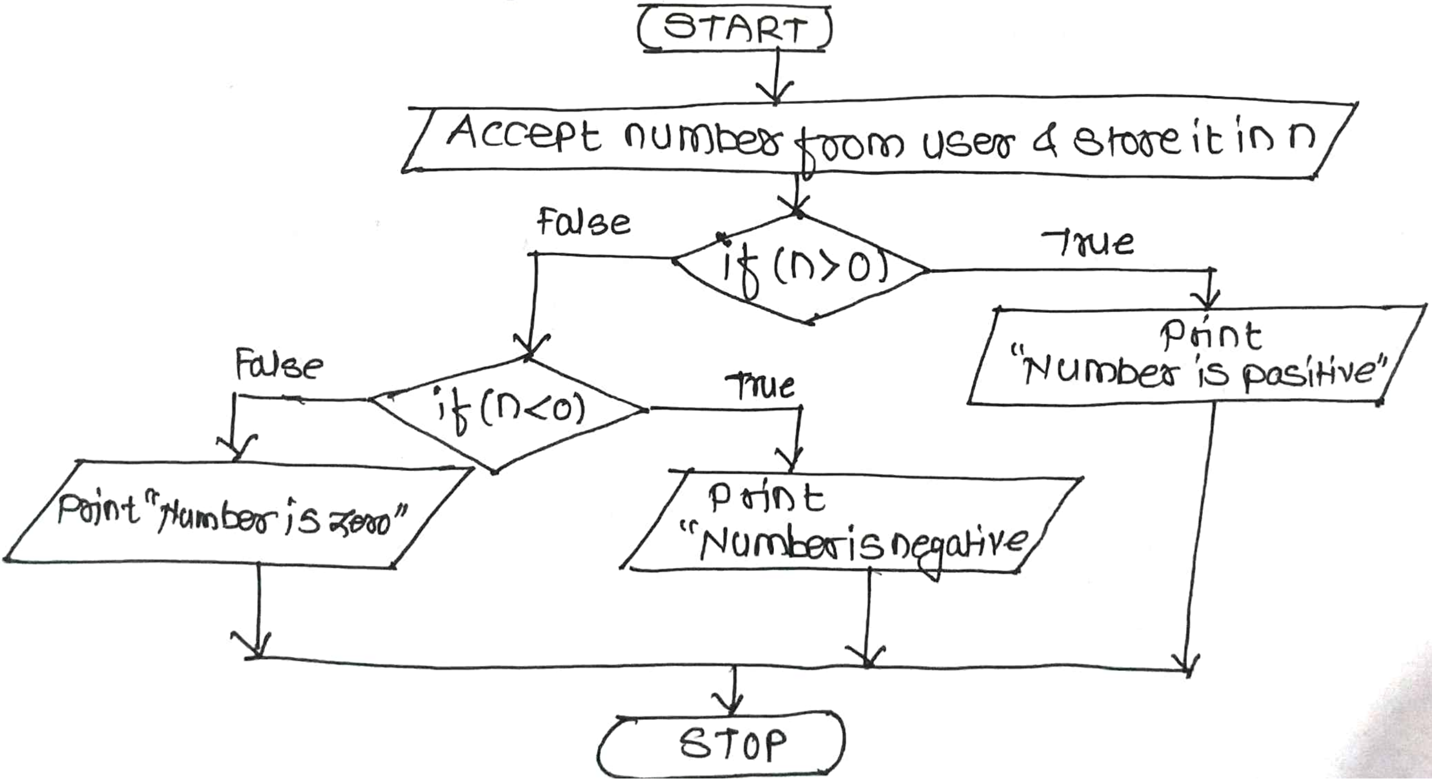
Flowchart of else if ladder :-



Program:- Write a program to accept a number from user and check whether it is positive, negative or zero.

or

Write suitable ~~of~~ example (program), explain how else...if ladder can be used.

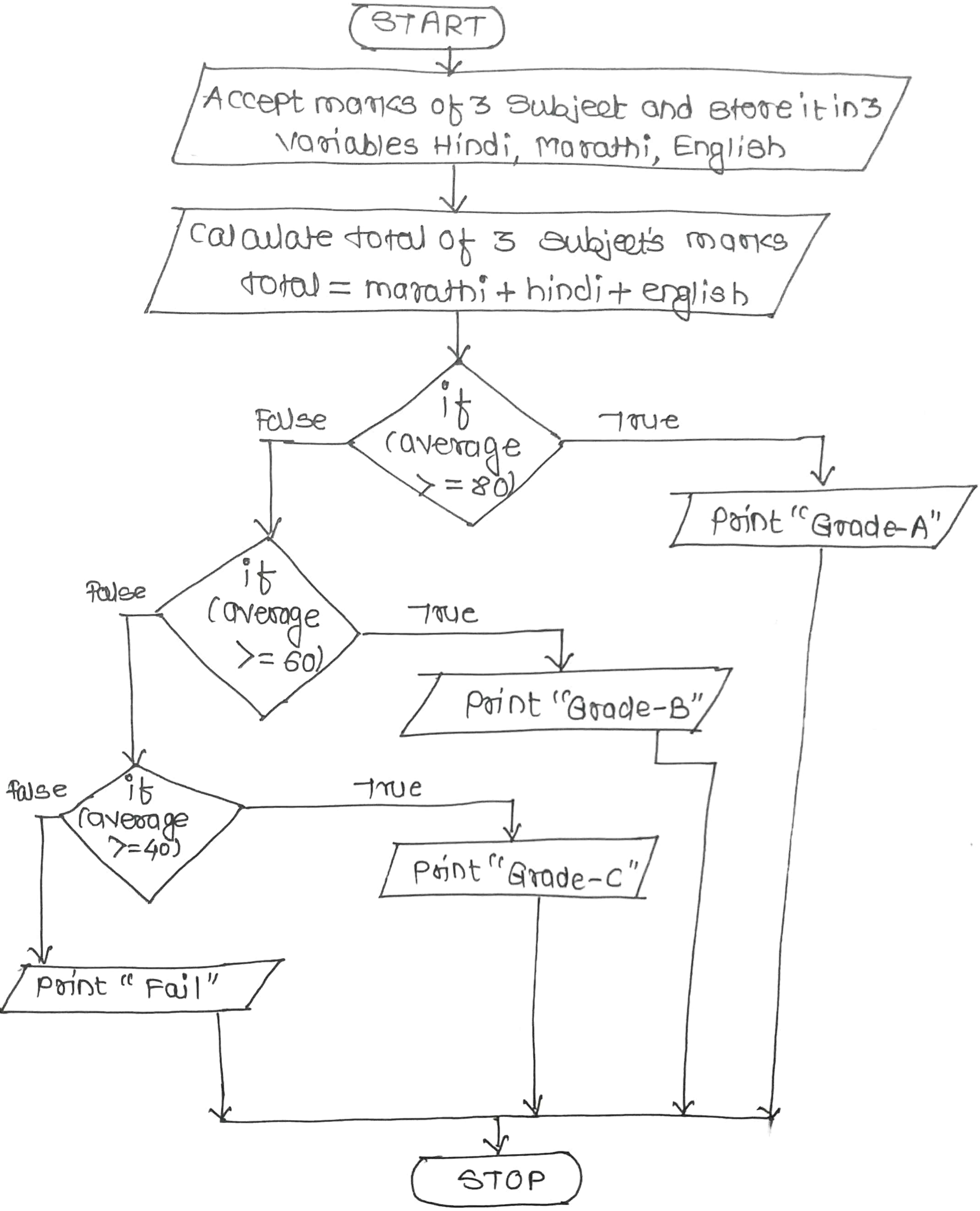

START
Accept number from user & store it in n
False
if (n>0)
True
Print "Number is positive"
False
if (n<0)
True
Print "Number is zero"
Print "Number is negative
STOP

```
#include <stdio.h>
#include <conio.h>
 main()
 {
  int n;
  printf("\n Enter a number:");
  scanf("%d", &n);
   if(n>0)
    {
     printf("\n Number is positive");
    }
   else if (n<0)
    {
     printf("\n Number is negative");
    }
  else
    {
      printf("\n Number is zero.");
    }
 }
```

Output:-

```
Enter a number : 5
Number is positive.
```

* Program:- Write a program to accept marks of 3 subjects from student. calculate the total & average of marks. If average is >=80 then give the grade as "A", if average is >= 60 then give the grade "B", if average >=40 then give the grade as "C" and below 40 "Fail".


START
Accept marks of 3 Subject and store it in 3 variables Hindi, Marathi, English
Calculate total of 3 subject's marks
total = marathi + hindi + english
if (average >= 80)
False
True
Print "Grade-A"
if (average >= 60)
False
True
Print "Grade-B"
if (average >= 40)
false
True
Print "Grade-C"
Print " Fail"
STOP

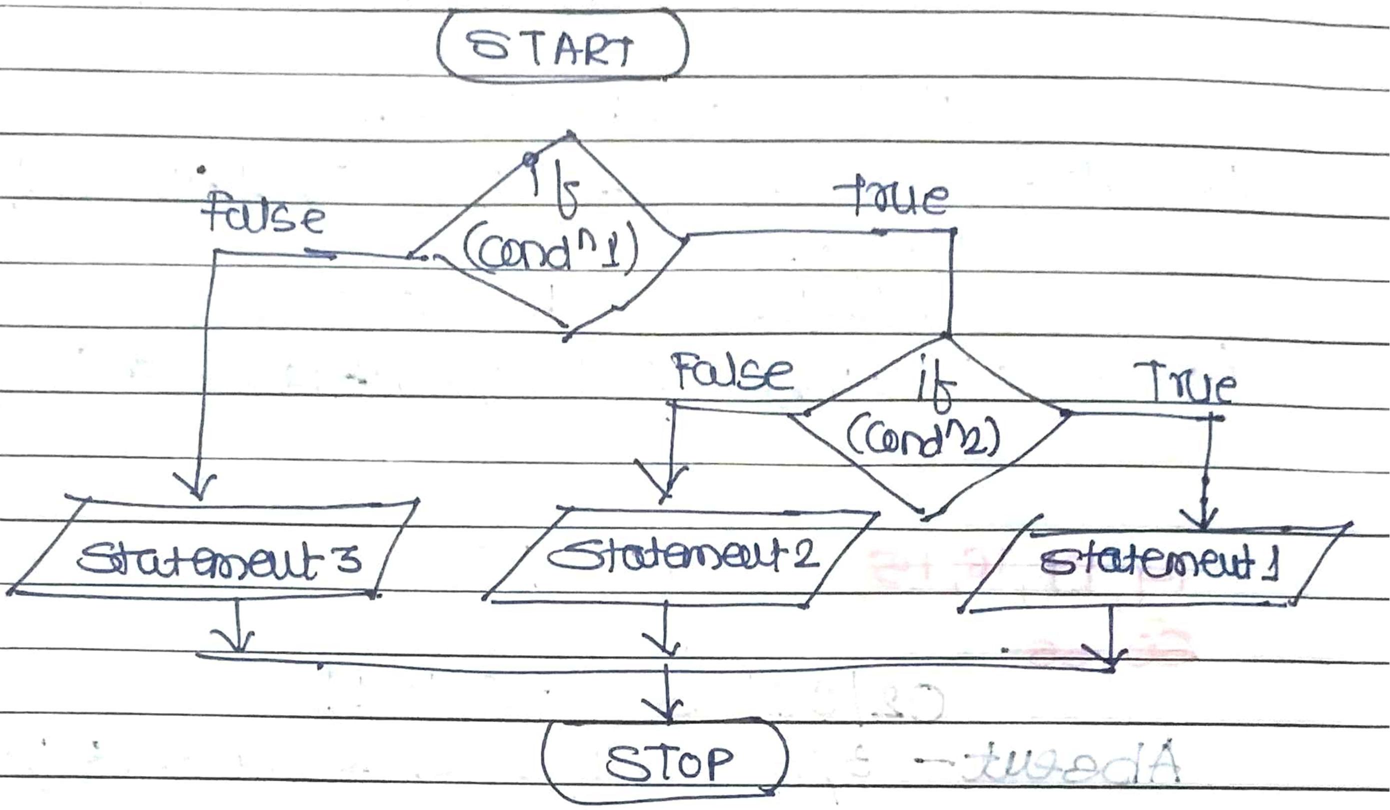
```
#include <stdio.h>
#include <conio.h>
main () {
int hindi, marathi, english, total, average;
printf("\n Enter marks of marathi, english and hindi: );
scanf("%d %d %d", &marathi, &english, &hindi);

total = marathi + english + hindi;
 average = total/3;
 if (average >= 80)
  {
   printf("\n Grade-A");
  }
 else if (average >= 60)
    {
    printf("\n Grade-B");
    }
 else if (average >= 40);
    {
     printf("\n Grade-C");
     }
  else
    {
     printf("\n Fail...");
    }
 }
```

Output:-

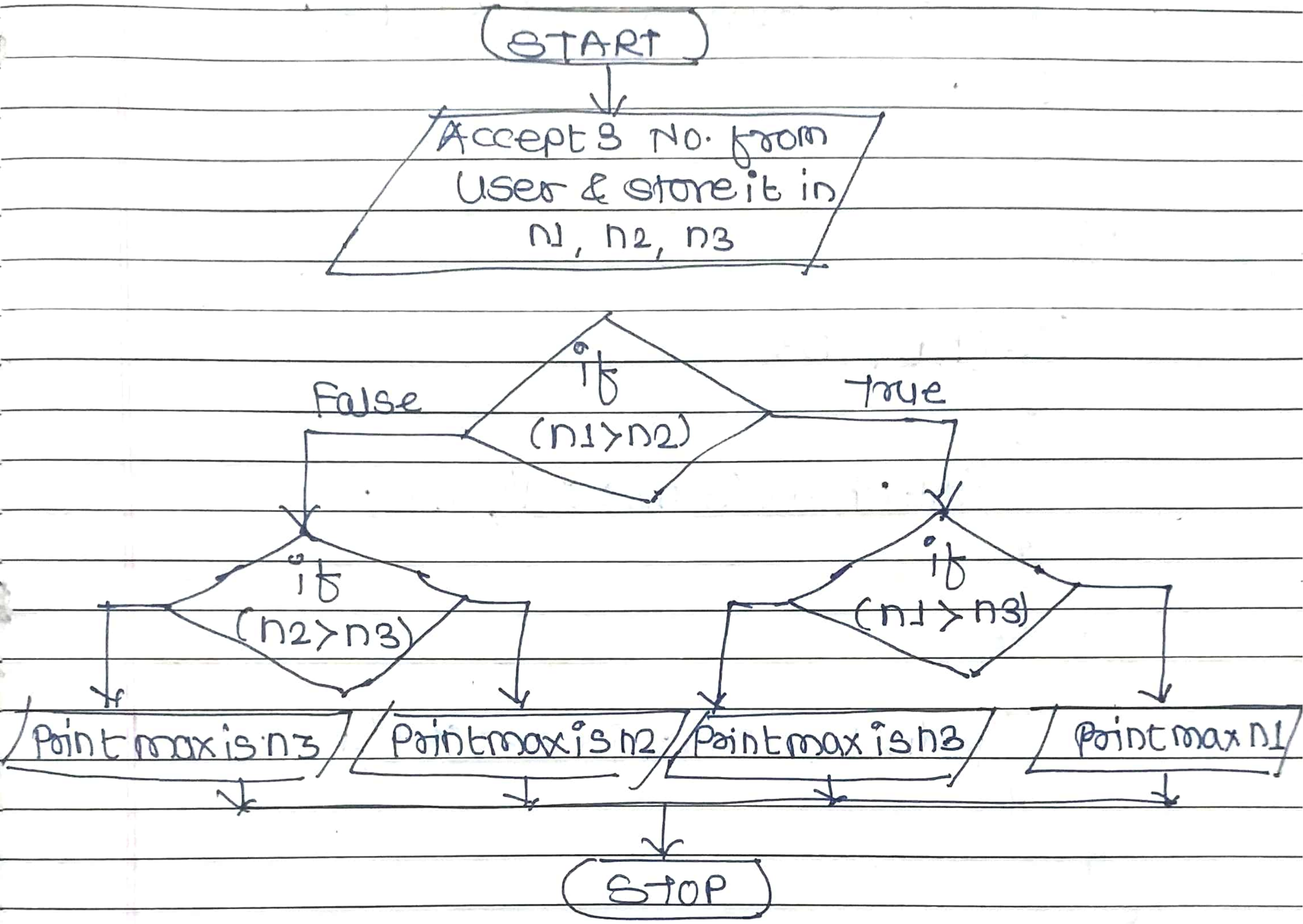
```
Enter marks of marathi, english, and hindi: 90 80 70
 Grade-A
```

* Nested if else Statement



```
if (condⁿ1)
 {
    if (condⁿ2)
    {
       Statement1;
    }
   else
    {
       statement2;
    }
 }
else
 {
    Statement 3;
 }
```

Write prog. to select & print the largest of the no. using nested-if-else statement

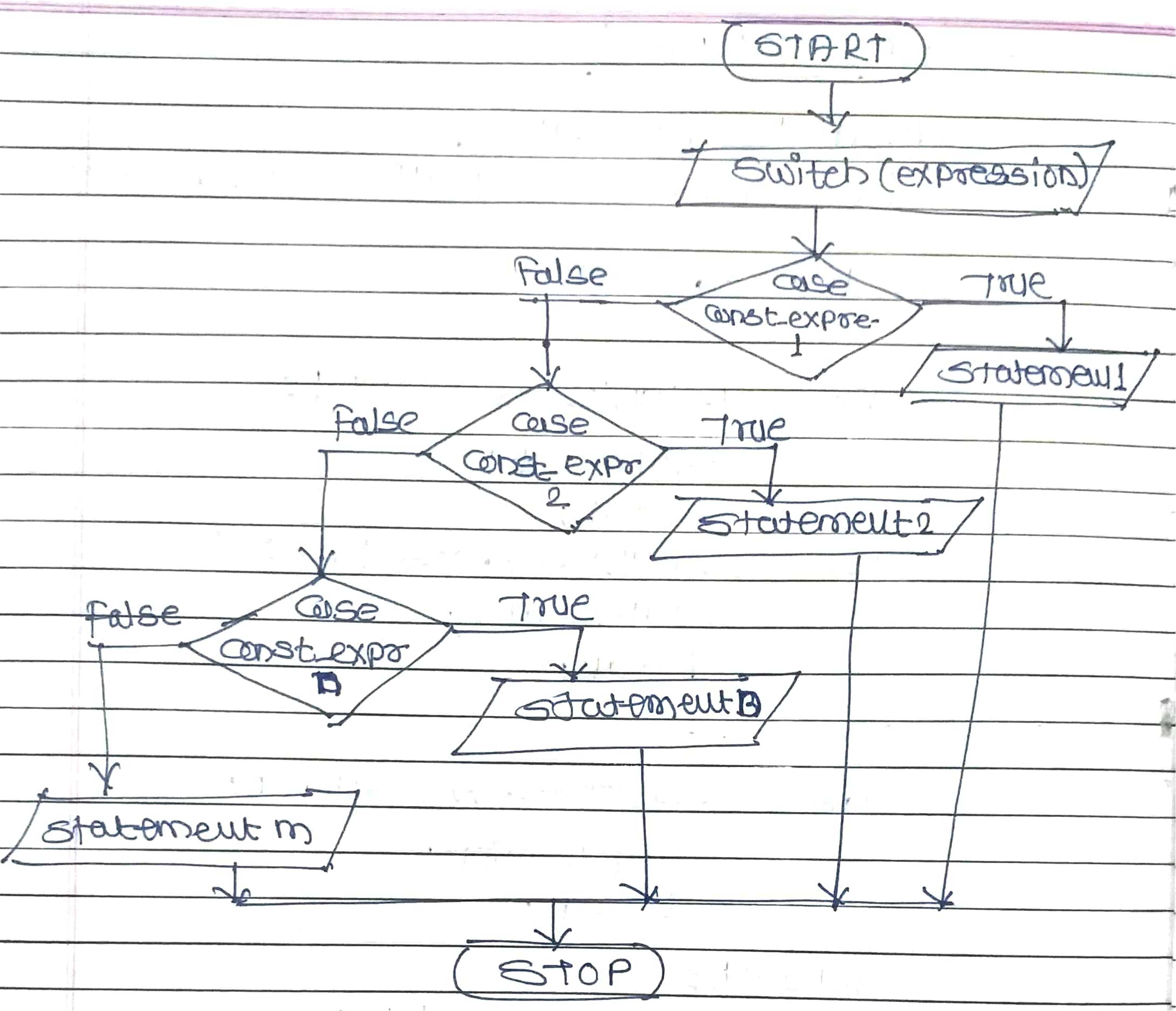


```
main()
{
  int n1, n2, n3;
   printf(" Enter 3 no.s ");
   scanf("%d %d %d", &n1, &n2, &n3);
   if(n1>n2)
    {
      if (n1>n3)
       { printf("max is n1 %d", n1);
       }
  else
   {
    printf ("max is n3: %d", n3);
   }
 }
else
 {
  if(n2> n3)
   {
    print('max is n2: %d", n2);
   }
 else
   {
     printf( "Max is n3: %d", n3);
   }
 }
}
```

Switch statement

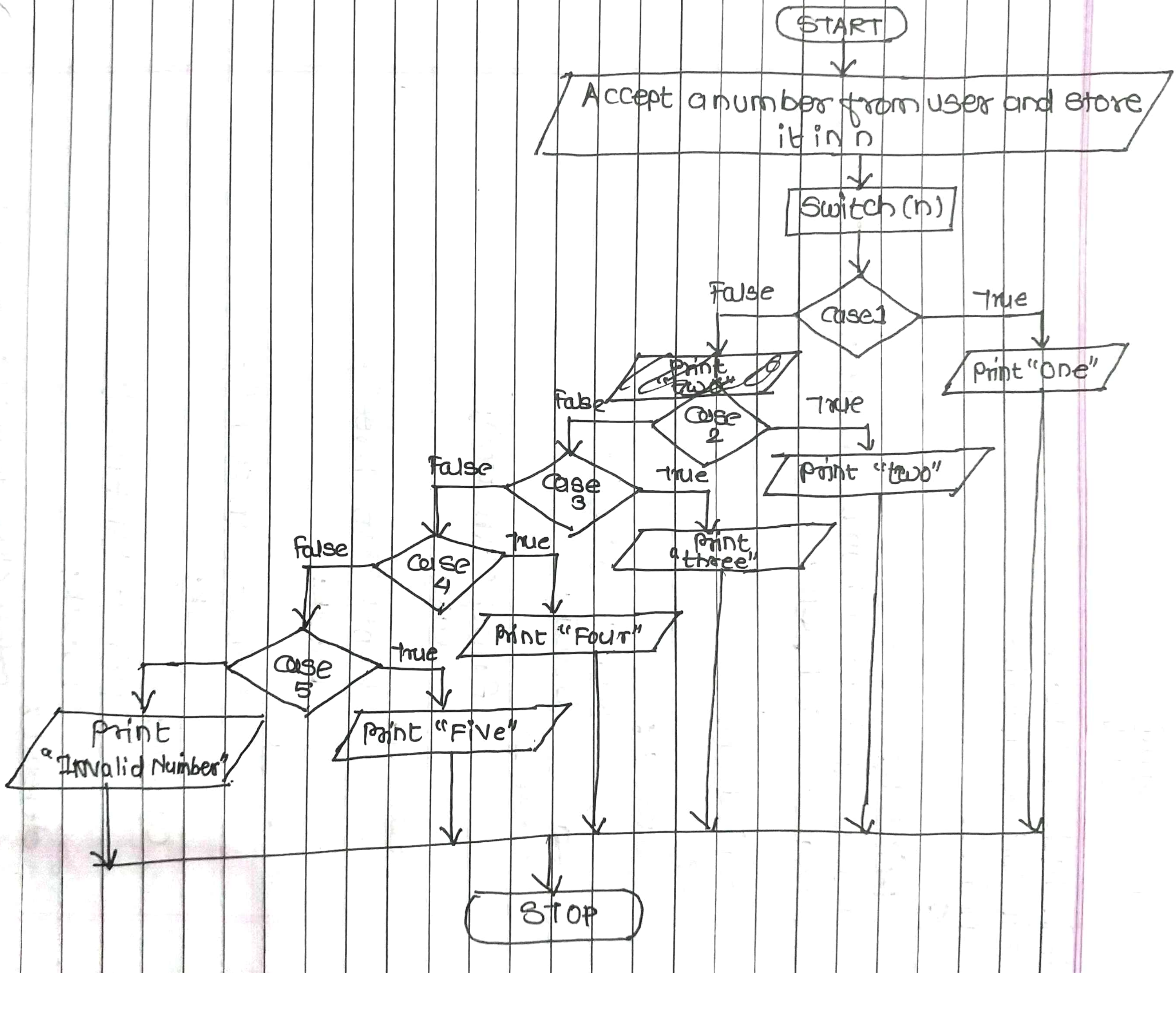
multi decision making
single option from multiple given option.

```
switch(expression)
{
 case constant_expression1:
            statement 1;
            break;
 case constant_expression2:
            statement 2;
            break;
  -----.
  case constant_expression n:
            statement n;
            break;
  default:
            statement m;
}
```



Prog8- Write a prog. to accept a number in the range 1 to 5 & print it in words.

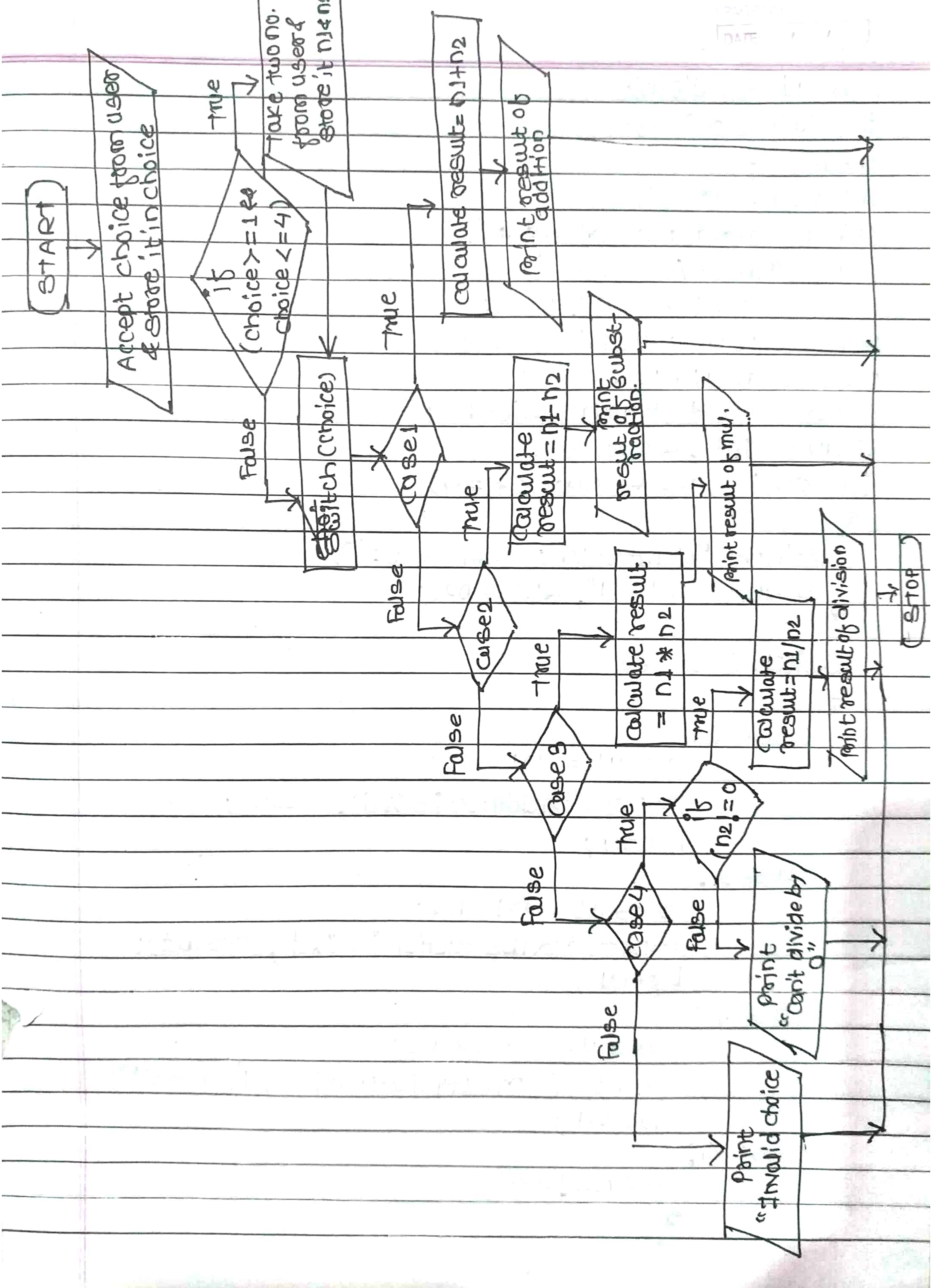
```
switch(n)
{
 case1:
          printf("In one");
          break;
case2:
default   printf(" Invalid Number");
```


START
Accept a number from user and store it in n
Switch (n)
False
Case1
True
Print "One"
~~Print "Two"~~
False
Case 2
True
Print "two"
False
Case 3
True
Print "three"
False
Case 4
True
Print "Four"
Case 5
True
Print "Five"
Print "Invalid Number"
STOP

```
#include <stdio.h>
#include<conio.h>
 main()
  {
   int n;
  printf("\n Enter a number between 1 to5:");
   scanf ("%d", &n);

   switch(n)
    {
     case 1:
                 printf("\none");
                 break;
     case 2:
                 Printf("\n two");
                 break;
     case 3:
                 Printf("\n three");
                 break;
     case 4:
                 Printf ("\n Four");
                 break;
     case 5:
                 Printf("\nFive");
                 break;
     default:
                 Printf("\n Invalid Number");
      }
   }

 Enter a number between 1 to 5:3
     three
```


START
Accept choice from user & store it in choice
if (choice>=1 && choice<=4)
True
Take two no. from user & store it n1 & n2
False
switch(choice)
Case1
True
Calculate result= n1+n2
Print result of addition
False
Case2
True
Calculate result=n1-n2
Print result of substraction.
False
Case3
True
Calculate result = n1*n2
Print result of mul.
False
Case4
True
if (n2!=0)
True
Calculate result=n1/n2
Print result of division
False
Print "can't divide by 0"
False
Print "Invalid choice"
STOP

```
#include<stdio.h>
#include<conio.h>
main(){
int n1, n2, result, choice;
printf("\n***MENU***");
printf("\n 1: Addition");
printf("\n 2: Substraction");
printf("\n 3: Multiplication");
printf("\n 4: Division");
printf("\n Select your choice: ");
scanf("%d", &choice);
if (choice>=1 && choice<=4)
 {
    printf("\nEnter two numbers: ");
     scanf("%d %d", &n1, &n2);
 }
 switch (choice)
 {
    case 1:
              result = n1+n2;
              printf("Addition is %d", result);
              break;
    case 2:
              result = n1-n2;
             printf("substraction is %d", result);
             break;
    case 3:
              result = n1*n2;
              printf("Multiplication is %d", result);
              break;
    case 4:
             if (n2!=0)
             {
```

```
            result = n1/n2;
            printf("\n Division is %d", result);
          }
        else
          {
            printf("\n Cannot divide by zero");
          }
        break;
    default:
            printf("\n Invalid Choice");
      }
  }
```

```
*** MENU ***
1: Addition
2: Substraction
3: Multiplication
4: Division
Select Your Choice : 1
 Enter two numbers : 10  5 ↵

 Addition is 15.
```

* Differences between if and switch statements

Parameters	if else	switch case
No. of curly brackets	Multiple curly brackets are used.	single curly bracket is used.
Impact on code if no. of condition increases	code becomes complicated when conditions increases.	code does not become complicated even if conditions increases.
Use of break statement.	Donot use break statements.	Use break statements.
Values used in condition	Tentative values can be given in condition eg. >= 80.	Exact values are expected in the conditions.

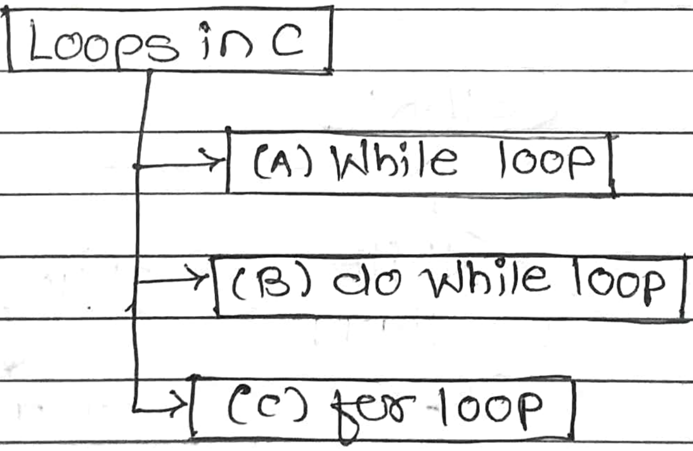
Syntax	if (condition)
	{
	statement
	}
	else
	{
	statement
	}
		switch (expression)
		{
		case const_expr:
		statements;
		break;
		----- 	
		~~break~~;
		default:
		statements;
		}

* Looping Statements :-

Use :-

- Loops are used to execute specific task repeatedly in our program. Rather than writting the code again and again we use the concept of loop.
- There are various situations when we may want to execute specific task multiple time. For example student marksheet.
- Here we want to accept details from student and want to generate the marksheet. This task is obviously repeated for number of students. In such situations we use the loops.

Looping Statement in C



☆ While loop :-

Use :-

- While loop is used when we want to execute the set of statements repeatedly until the given condition is satisfying.
- While loop is considered as an entry controlled loop. That means the condition

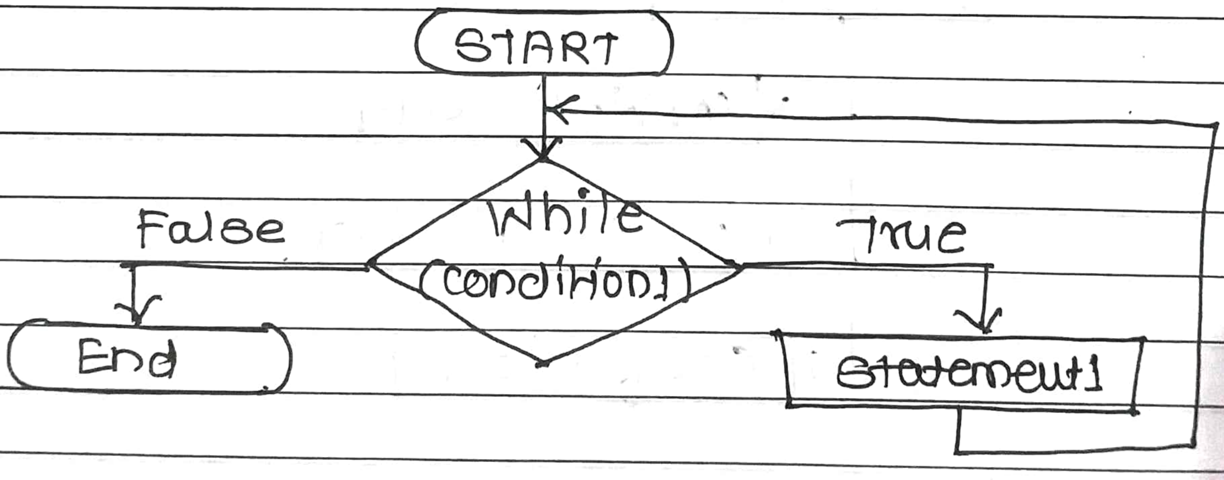
is given at the begining of loop. If the given condition does not satisfy, the loop statements never get executed.

- If the ~~statement~~ condition is satisfied, then loop statements are repeatedly executed until the condition is satisfying. Once the condition becomes false, the control exist the loop.

Syntax of While loop:-

```
While (condition1)
{
   statement 1;
}
```

Flowchart of While loop:-

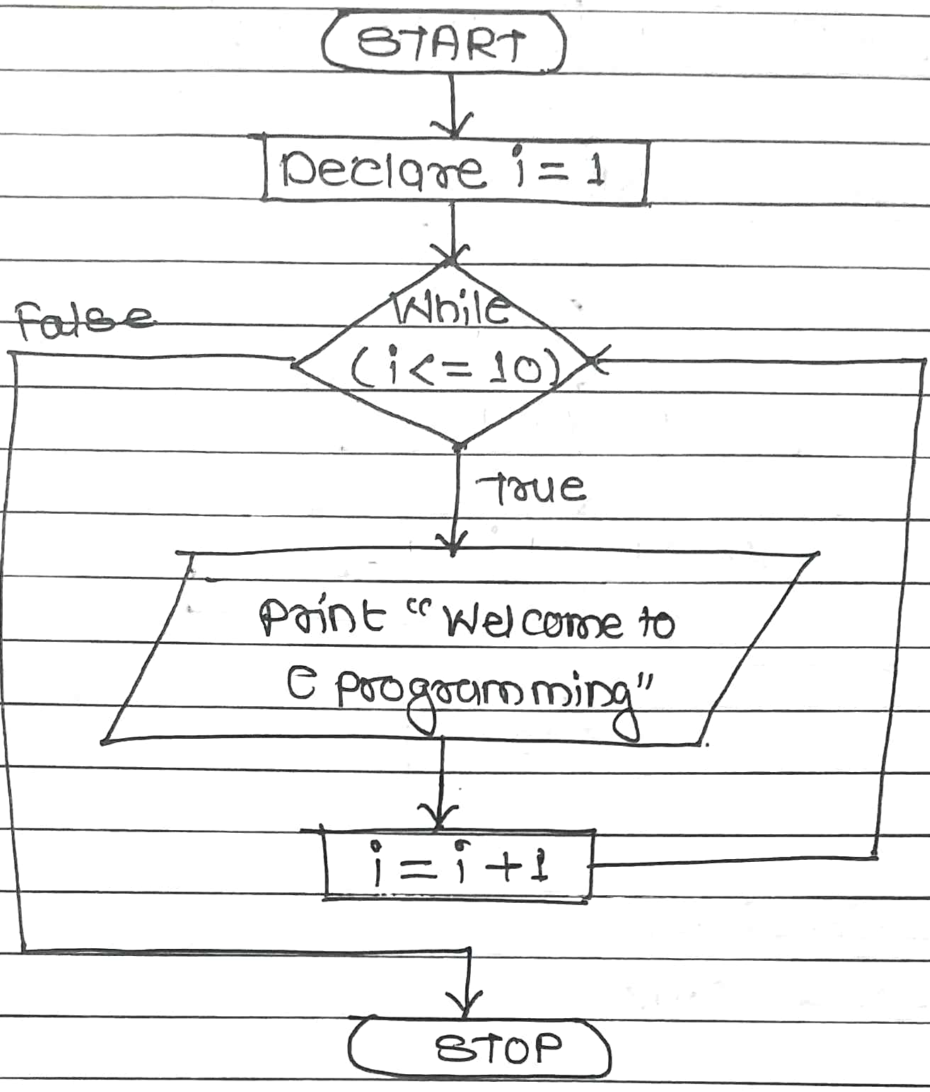


☆ Program:-

Write ~~the~~ a program to print a message 10 times.

⇒

Flowchart

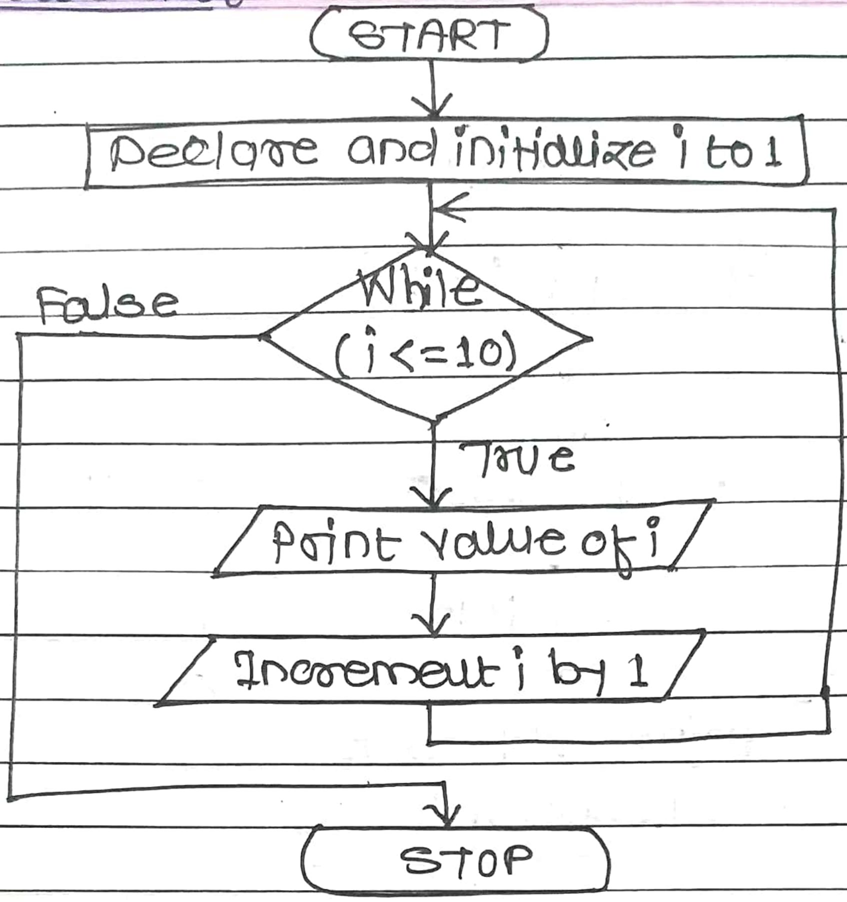


Program:-

```
#include <stdio.h>
#include <conio.h>
 main()
 {
    int i;
    i=1;
    While (i<=10)
     {
        printf("\n Welcome to C programming.");
        i=i+1;
     }
 }
```

Program:- Write a program to print 1 to 10 numbers.

Sol^n ⇒ Flowchart:-



Program:-

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

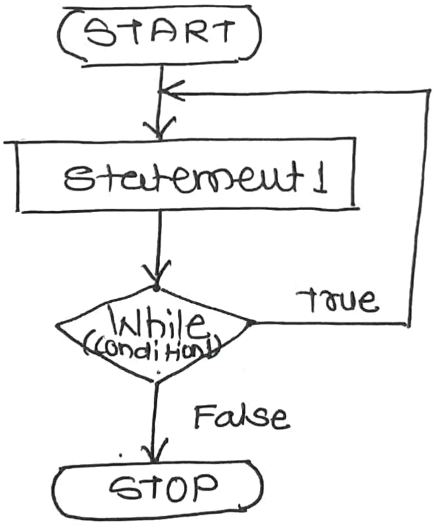
* do while loop :-

The do while loop is exit controlled loop or bottom tested loop. That is the condition is checked at the end of loop. Hence even if the condition does not satisfy, the loop statements will be executed at least once.

Syntax of do while loop :-

```
do
 {
      statement 1;
  } while (condition 1);
```

Flowchart :-



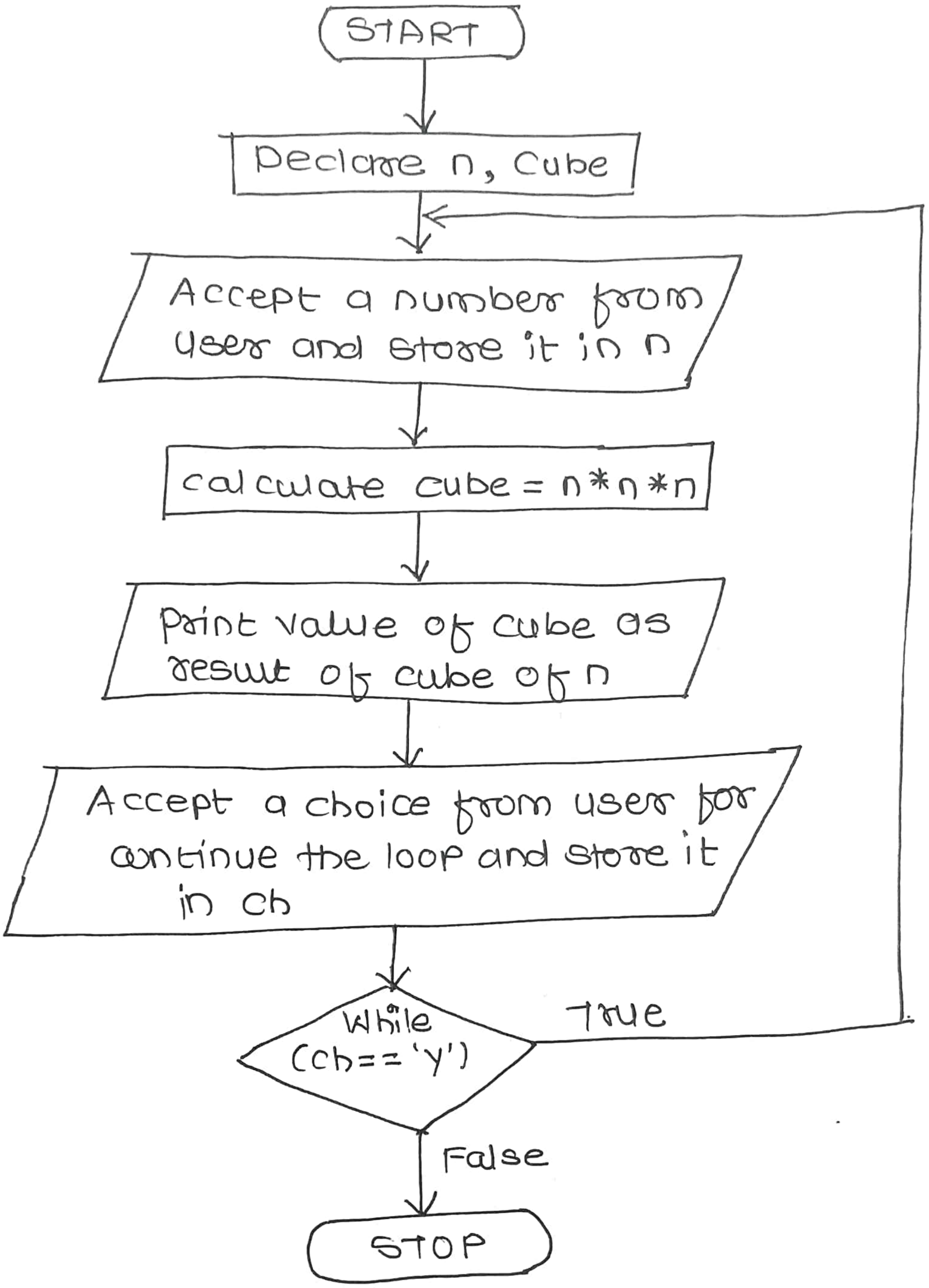
Program :-

Write a program to accept a number from user and print it's cube. Ask user for continuity, if user says yes repeat the process.

OR

State example (program) of do while loop.

Flowchart:-

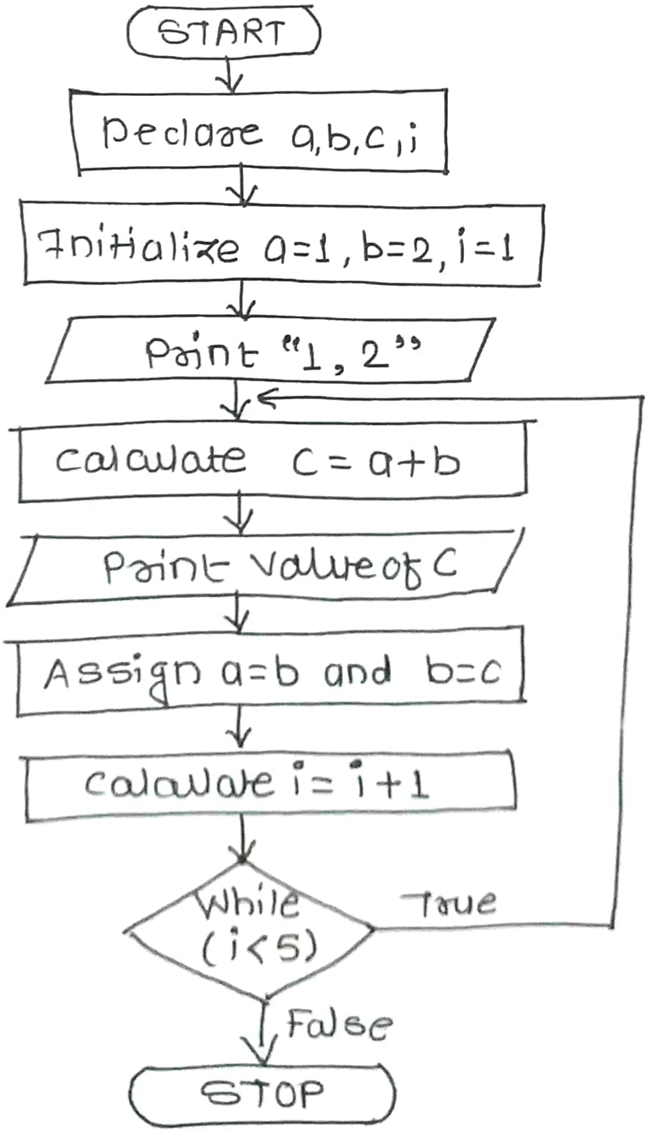

START
Declare n, Cube
Accept a number from user and store it in n
calculate cube = n*n*n
Print value of cube as result of cube of n
Accept a choice from user for continue the loop and store it in ch
While (ch=='y')
True
False
STOP

Program:-

```
#include <stdio.h>
#include <conio.h>
void main()
 {
   int n, cube; char ch;
do
{
    printf("\n Enter a number : ");
    scanf( "%d", &n);
    cube = n*n*n;
    printf("\n Cube is %d", cube);
    printf("\n\n Do you want to continue? (y/n):");
    scanf ("%s", &ch);
  } while (ch=='y');

  getch();
}
```

* Write a program to print Fibonacci series.



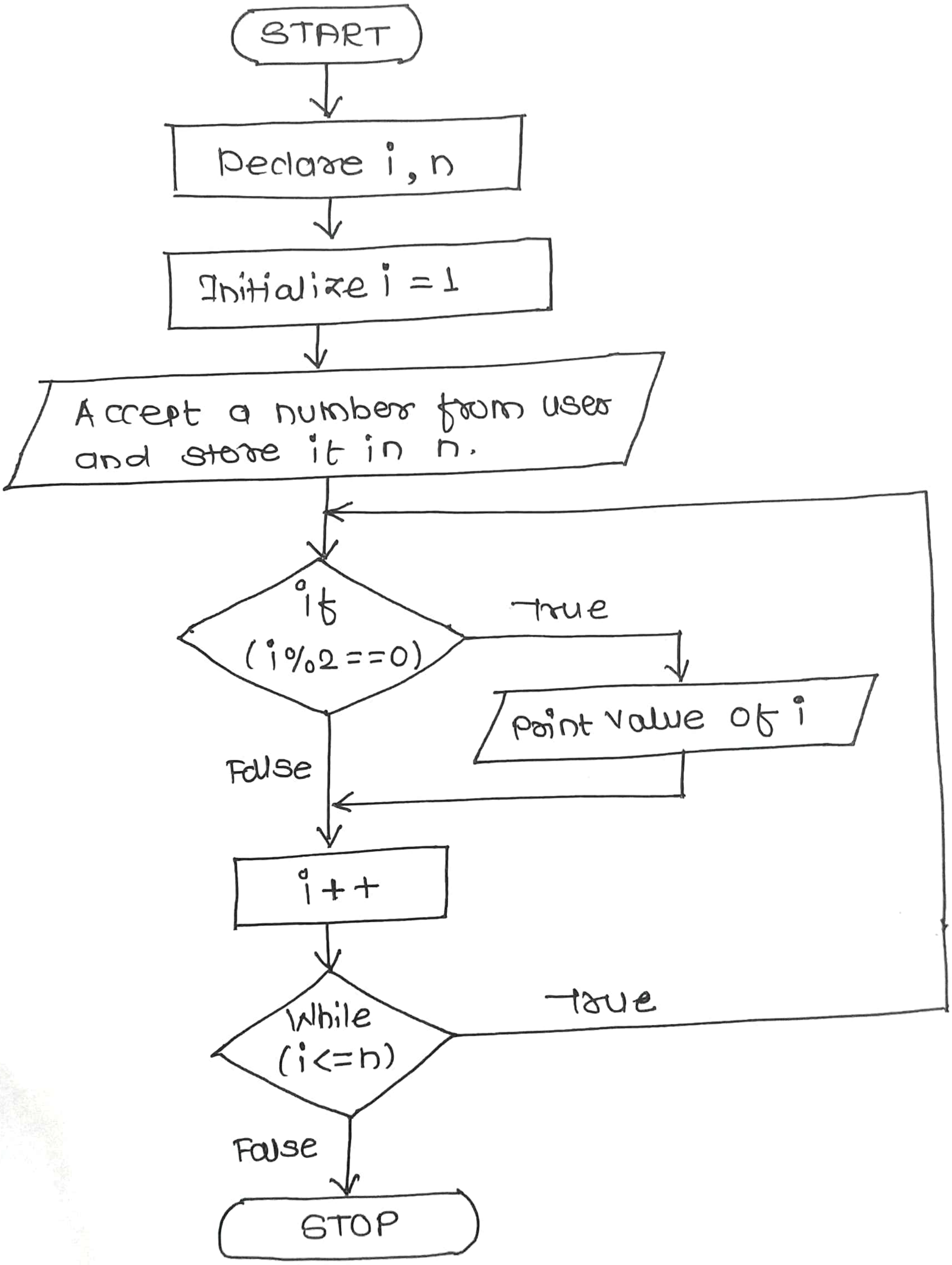
```
#include <stdio.h>
#include <conio.h>
 void main ()
  {
     int a, b, c, i;
     a=1;
     b=2;
     i=1;
     Print ("1 2");
     do
      {
         c = a+b
         Print ("%d", c);
         a=b;
         b=c;
         i=i+1;
      } while (i<5);
     getch ();
  }
```

	While (i<5)	i	a	b	c	O/P on screen
Initially		1	1	2	–	1 2
Iteration 1	While (1<5) is true	2	2	3	3	1 2 3
Iteration 2	While (2<5) is true	3	3	5	5	1 2 3 5
Iteration 3	While (3<5) is true	4	5	8	8	1 2 3 5 8
Iteration 4	While (4<5) is true	5	8	13	13	1 2 3 5 8 13
Finally	While (5<5) is false					

* Write a program to display all even numbers from 1 to N.

⇒

Flowchart:-



Program :-

```
#include <stdio.h>
#include <conio.h>
 void main ()
  {
      int i, n;
      i = 1;
      Printf ("\n Enter value for n: ");
      scanf ( "%d", &n);
       do
        {
             if (i%2 == 0)
                 Printf("%d", i);
                 i++;
        } while (i<=n);
  }
```

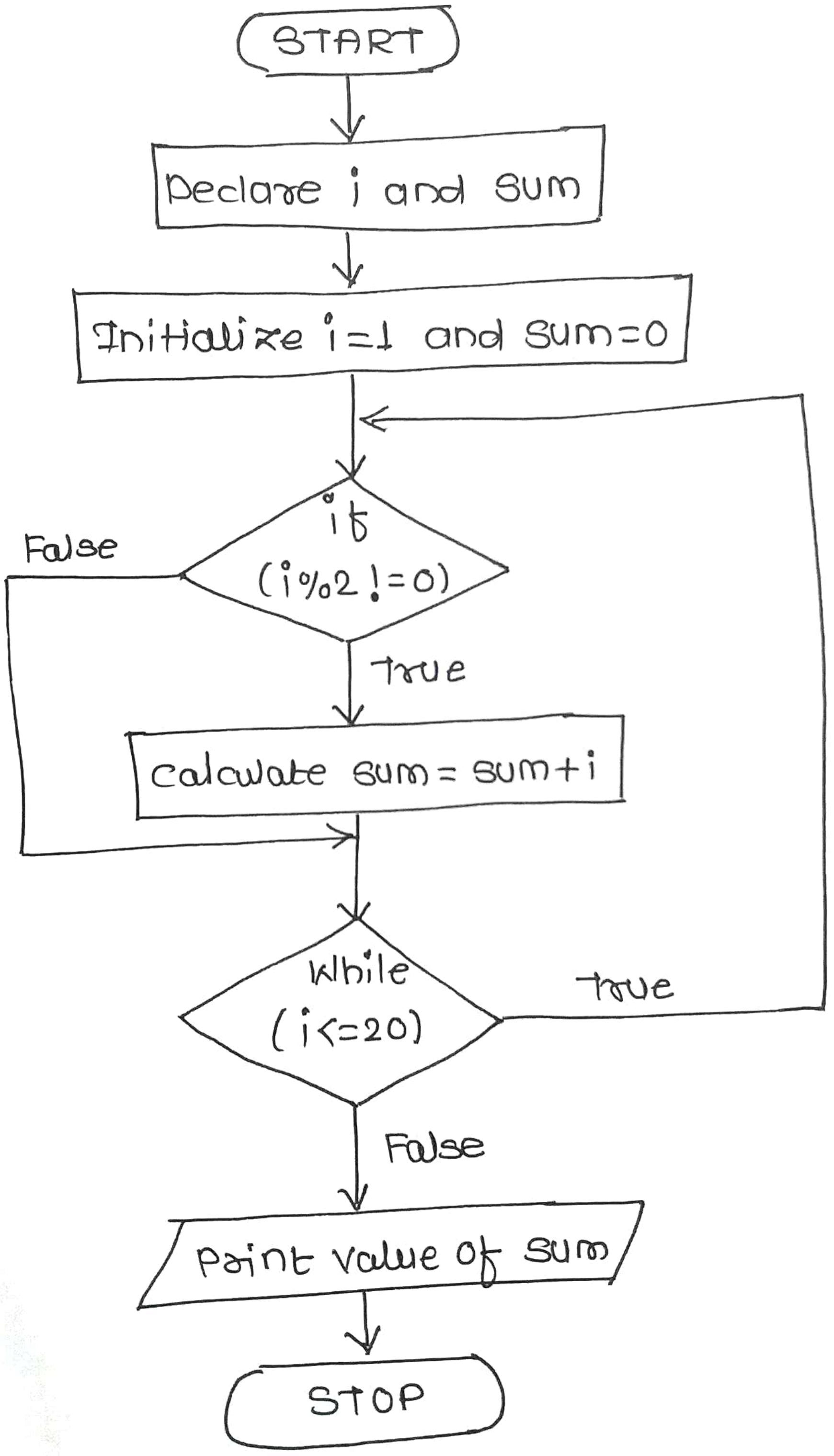
Output :-

```
Enter value for n: 8
 2  4  6  8
```

* Write a program to find the sum of first N odd numbers.

⟹

Flowchart :-



Program:

```
#include <stdio.h>
#include <conio.h>
void main ()
  {
     int i, sum, N;
     i=1;
     sum=0;
     N=20;
     do
      {
         if (i%2 !=0)
              sum= sum+i;
              i=i+1;
      } while (i<=N);

     printf("\n sum of odd numbers: %d", sum);
     getch();
  }
```

20 times loop will be executed

Output:-

Sum of odd numbers: 100.

Explanation:-

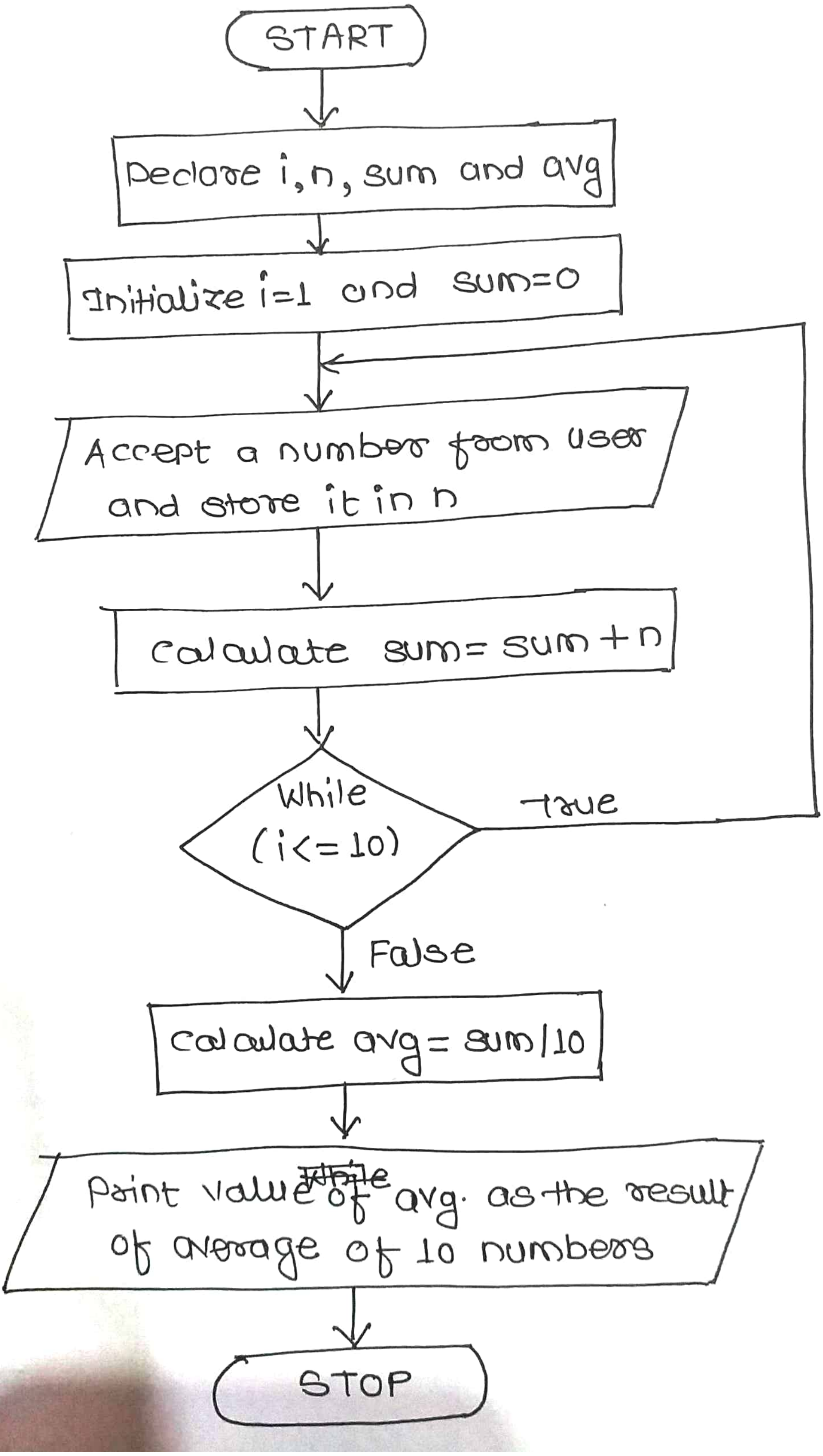
It will print addition of 1+3+5+7+9+11+13+15+17+19 which is 100.

* Write a program to accept ten numbers and print average of it.

Sol

Solⁿ ⇨

Flowchart:-


START
Declare i, n, sum and avg
Initialize i=1 and sum=0
Accept a number from user and store it in n
Calculate sum=sum+n
While (i<=10)
True
False
Calculate avg=sum/10
Print value ~~While~~ of avg. as the result of average of 10 numbers
STOP

Program:-

```
#include <stdio.h>
#include<conio.h>
 void main ()
  {
     int i, n, sum, avg;
     i=1;
      sum= 0;
       do
       {
            printf("\n Enter number: ");
            scanf("%d", &n);
             sum= sum+n;
              i=i++;
        } While (i<=10);
        avg= sum/10;
        printf("\n Average is %d", avg);
        getch();
    }
```

output:-

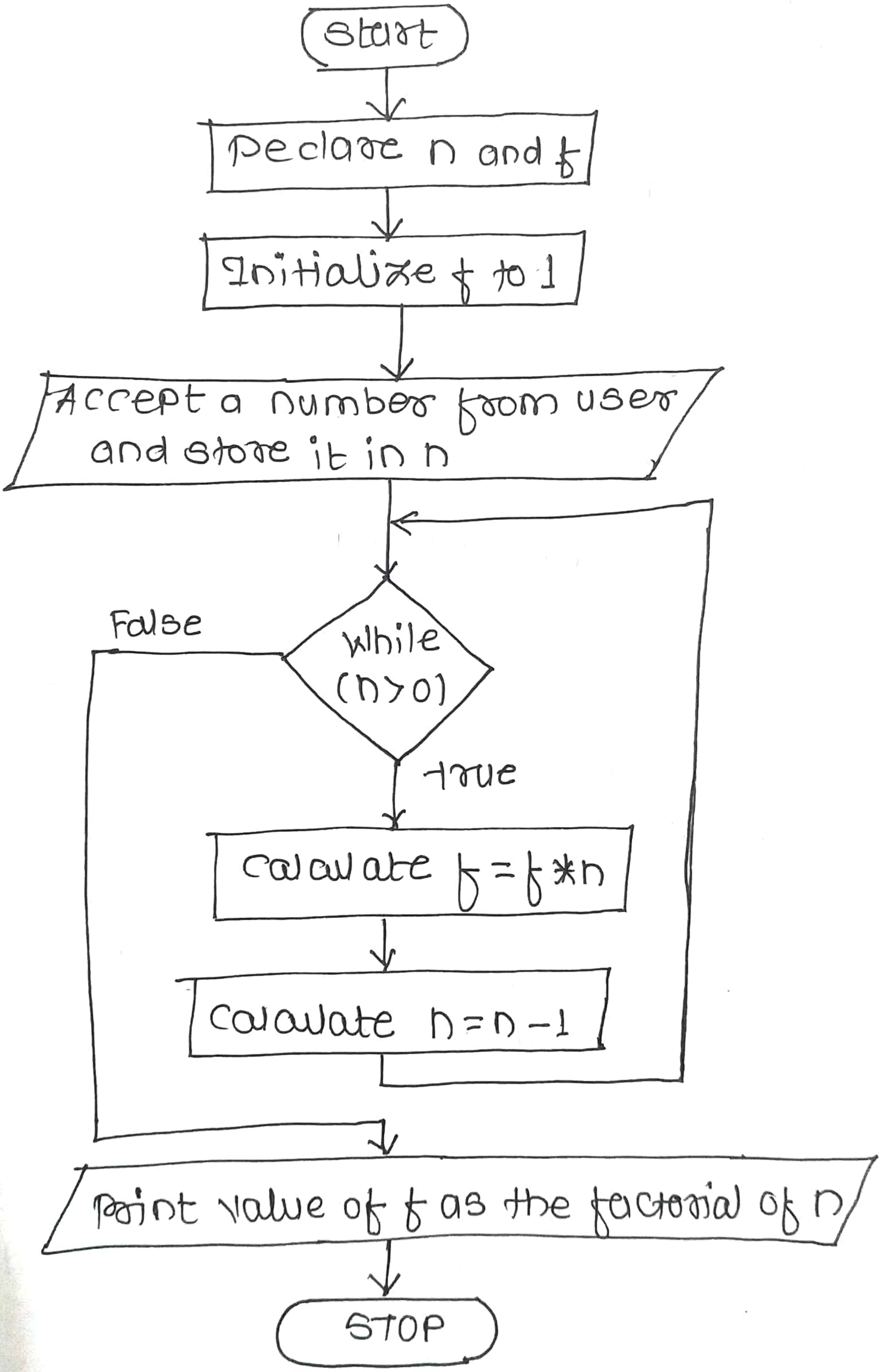
```
Enter number: 1
Enter number: 2
Enter number: 3
Enter number: 4
Enter number: 5
Enter number: 6
Enter number: 7
Enter number: 8
Enter number: 9
Enter number: 10

Average is 5.
```

Program: Write a program to accept a number from user and print factorial of it.

e.g. Factorial of 5 is calculated as = 5*4*3*2*1=120

Flowchart :-



* Program:-

```
#include <stdio.h>
#include <conio.h>
main(){
 int n,f;
 f = 1;
 Printf("\n Enter a number:")
 Scanf("%d", &n);
  while (n>0)
   {
     f = f * n;
      n = n-1;
   }
 Printf("\n Factorial is %d.", f);
 }
```

Output:-

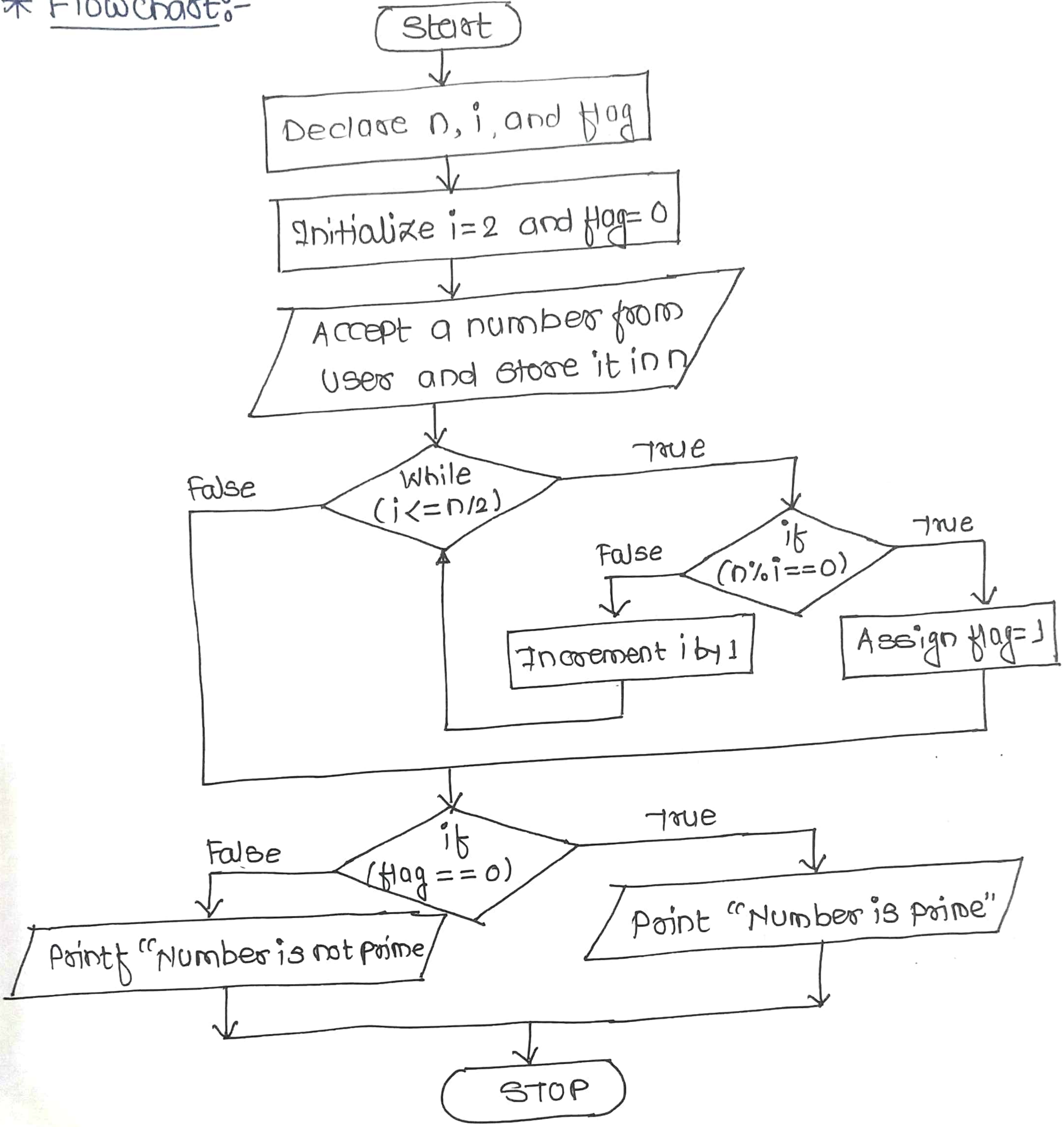
```
Enter a number: 5
Factorial is 120.
```

✲ Program: Write a program to find whether the entered number is prime or not.

Soln ⇒

Prime number is the one which is divisible by 1 and itself only.

* Flowchart:-



* Program:-

```
#include<stdio.h>
#include<conio.h>
 main() {
  int n,i, flag;
   i=2;
   flag=0;
```

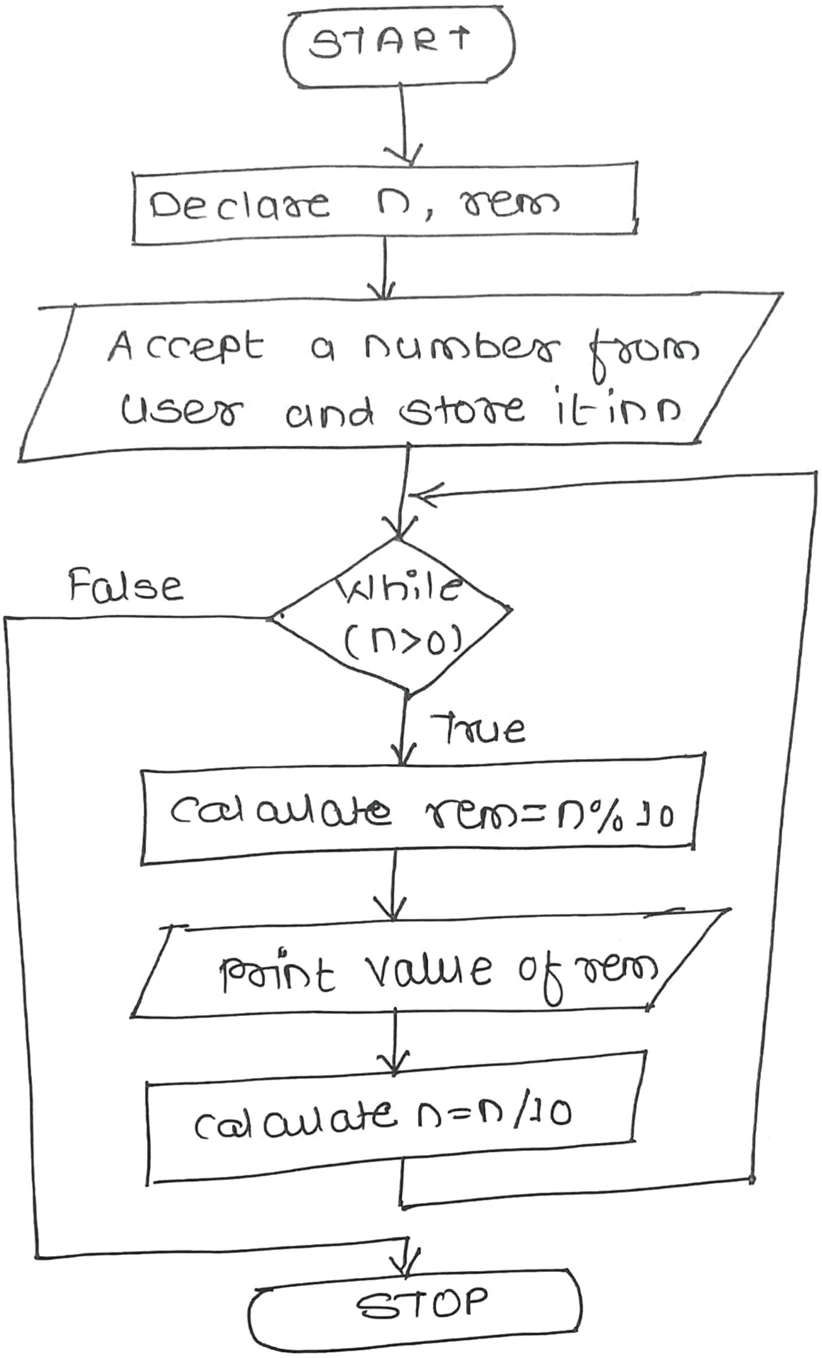
```
printf ("\n Enter a number:");
scanf( "%d", &n);
While (i<=n/2)
 {
    if(n%i==0)
     {
       flag=1;
       break;
     }
    i=i+1;
   if( flag==0)
    printf("\n Number is prime");
   else
    printf("\n Number is not prime");
 }
```

Output:-

```
 Enter a number: 11 ↵
  Number is prime
```

* Program:- Write a program to take input as a number and reverse it by using while loop.

⇨



Program:-

```
#include<stdio.h>
#include<conio.h>
void main ()
 {
     int n, rem;
     printf ("\n Enter a number:");
     scanf("%d", &n);
     printf(" \n Reverse number is:");
while (n>0)
 {
    rem = n%10;
    printf("%d", rem);
     n = ~~n/1~~0 n/10;
   }
 }
```