INDEX					
MODULE NO	Date	IDX	PROGRAM	PAGE NO	Sign & Marks
4		**Simple computational problems using the operator' precedence and associativity**			
		4.1	Evaluate the following expressions. a.A+B*C+(D*E) + F*G b. A/B*C-B+A*D/3 c.A+++B---A d. J= (i++) + (++i)		
		4.2	Find the maximum of three numbers using conditional operator		
		4.3	Find the given number is even or odd or zero		
		4.4	Find the given number is +ve or -ve or neutral		
		4.5	Find the grade of a student		
		4.6	Find the roots of a quadratic equation		
		4.7	Multiplication Table		
		4.8	Find the factorial of given number using any loop.		
		4.9	Factors		
		4.10	Perfect number or not		

MODULE 4 - Program 4.1

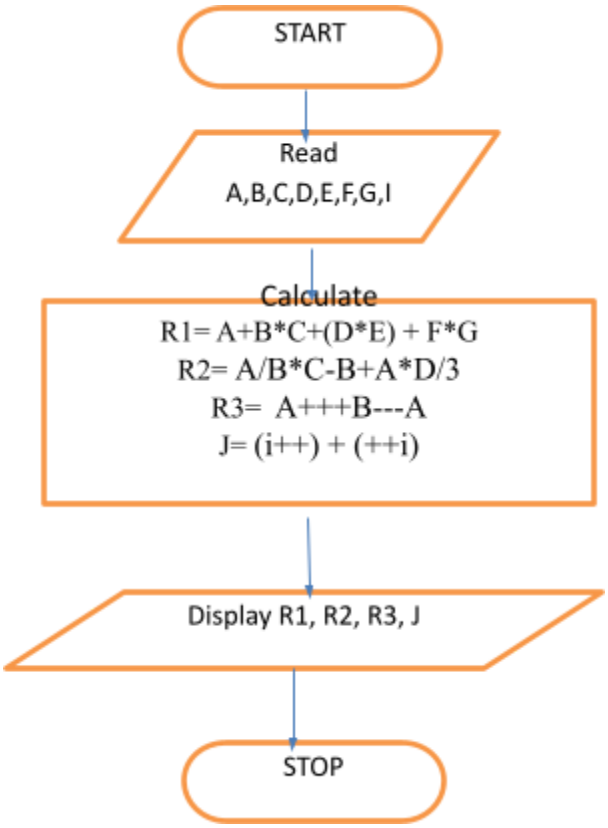
AIM : Write a C program to Evaluate the following expressions.

a) A+B*C+(D*E) + F*G
b) A/B*C-B+A*D/3
c) A+++B---A
d) J= (i++) + (++i)

ALGORITHM :

STEP 1 : START
STEP 2 : Read A, B, C, D, E, F, G, I values
STEP 3 : Calculate R1= A+B*C+(D*E) + F*G
STEP 4 : Calculate R2= A/B*C-B+A*D/3
STEP 5 : Calculate R3= A+++B---A
STEP 6 : Calculate J= (i++) + (++i)
STEP 7: Display R1, R2, R3, J
STEP 5 : Stop

FLOWCHART :



PROGRAM :

```
#include<stdio.h>
int main()
{
   int A, B, C, D, E, F, G, I,R1,R2,R3,J;
   printf("\n Enter the values of A, B, C, D, E, F, G, I : ");
   scanf("%d %d %d %d %d %d %d %d",&A, &B, &C, &D, &E, &F, &G, &I);
   R1= A+B*C+(D*E)+F*G;
   printf("\n Value of A+B*C+(D*E)+F*G = %d",R1);
   R2= A/B*C-B+A*D/3;
   printf("\n Value of A/B*C-B+A*D/3 = %d",R2);
   R3= A+++B---A;
   printf("\n Value of A+++B---A = %d",R3);
   J= (I++) + (++I);
   printf("\n Value of (i++) + (++i) = %d",J);

   return 0;
}
```

OUTPUT :

```
Enter the values of A, B, C, D, E, F, G, I : 1 2 3 4 5 6 7 8

Value of A+B*C+(D*E)+F*G = 69
Value of A/B*C-B+A*D/3 = -1
Value of A+++B---A = 1
Value of (i++) + (++i) = 18
```

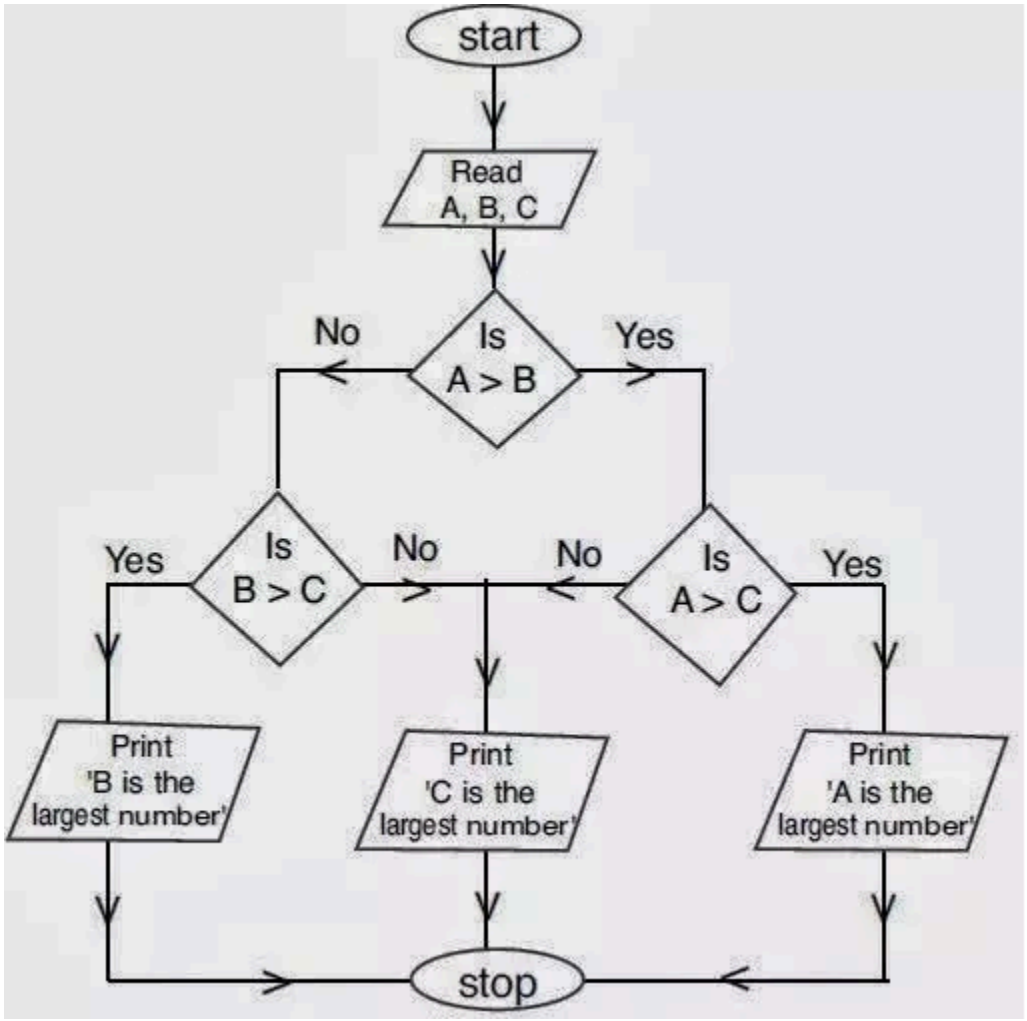
MODULE 4 – Program 4.2

AIM : write a C program to find the biggest of three numbers.

ALGORITHM :

STEP 1 : START
STEP 2 : Read a, b ,c values
STEP 3 : if (a>b) and (a>c) then display a is bigger go to step 5
STEP 4 : If b>c then display b is bigger otherwise display c is bigger
STEP 5 : Stop

FLOWCHART :



PROGRAM :

```c
#include<stdio.h>
 int main()
 {
  int a,b,c,big;
  printf("\n enter any three numbers\t");
  scanf("%d %d %d",&a,&b,&c);
  big=a>b?(a>c?a:c):(b>c?b:c);
  printf("Biggest number = %d",big);
  return 0;
 }
```

```
#include<stdio.h>
int main()
 {
  int a,b,c;
 printf("\n enter any three numbers\t");
  scanf("%d %d %d",&a,&b,&c);
  if ((a>b)&&(a>c))
          printf("%d is big ",a);
    else  if (b>c)
         printf("%d is big ",b);
      else
          printf("%d is big ",c);
     return 0;
 }
```

OUTPUT :

```
enter any three numbers        3 2 4
4 is big
 enter any three numbers        3 5 2
5 is big
 enter any three numbers        7 4 3
7 is big
```

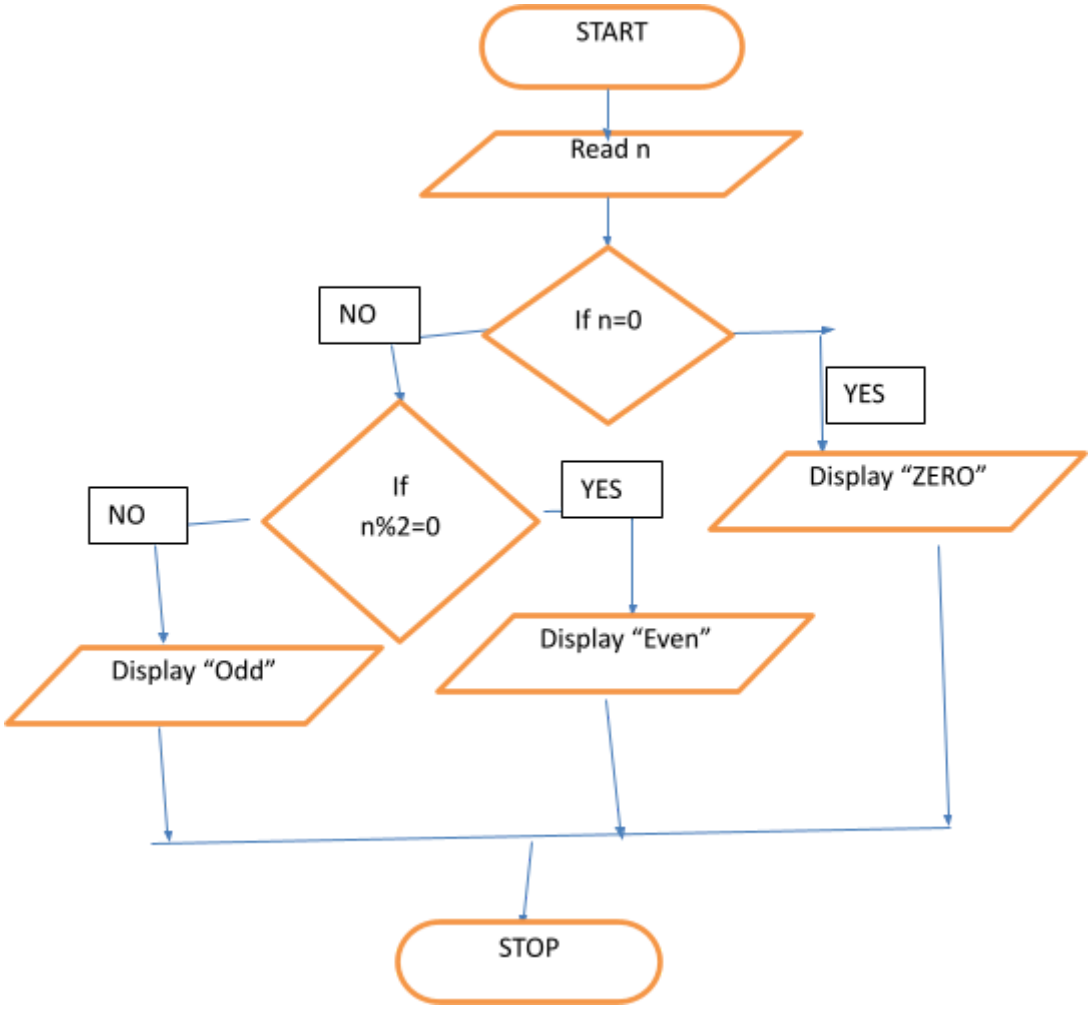
MODULE 4 - Program 4.3

AIM : Write a C program to find the given number is even or odd or zero

ALGORITHM :

STEP 1 : START
STEP 2 : Read n
STEP 3 : if n==0 then display "zero" goto step 5
STEP 4 : if n%2==0 then display " Even" otherwise display " Odd "
STEP 5 : Stop

FLOWCHART :



PROGRAM :

```
#include<stdio.h>
int main()
{
   int n;
   printf("\n Enter any number : ");
   scanf("%d",&n);
   if(n==0)
     printf("\n The given number is Zero ");
   else if(n%2==0)
     printf("\n The given number is Even ");
   else
     printf("\n The given number is Odd ");

   return 0;
}
```

OUTPUT :

```
Enter any number : 4

The given number is Even
```

```
Enter any number : 5

The given number is Odd
```

```
Enter any number : 0

The given number is Zero
```

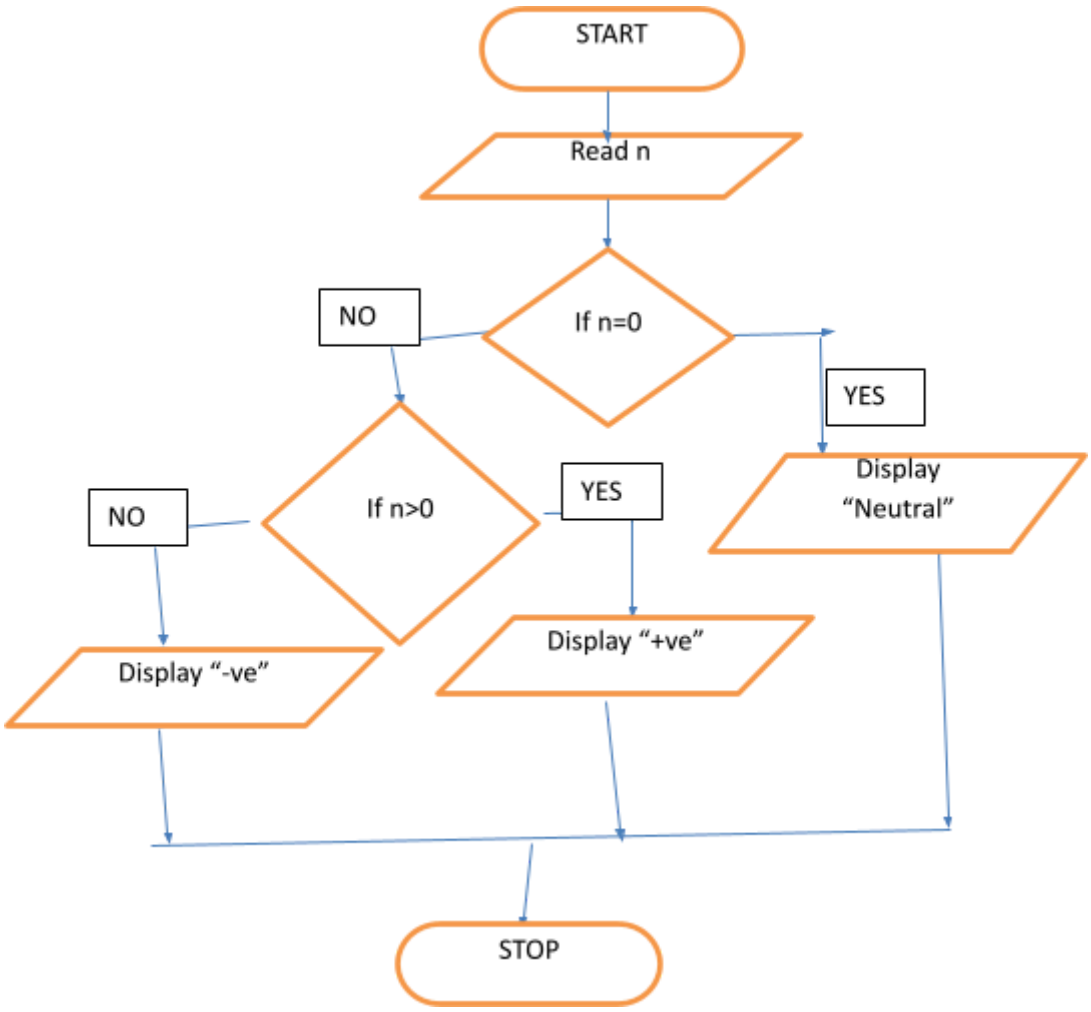
MODULE 4 - Program 4.4

AIM : Write a C program to find the given number is +ve or -ve or Neutral

ALGORITHM :

STEP 1 : START
STEP 2 : Read n
STEP 3 : if n==0 then display "Neutral" goto step 5
STEP 4 : if n>0 then display " +ve" otherwise display " -ve "
STEP 5 : Stop

FLOWCHART :



PROGRAM :

```
#include<stdio.h>
int main()
{
   int n;
   printf("\n Enter any number : ");
   scanf("%d",&n);
   if(n==0)
     printf("\n The given number is Neutral ");
   else if(n>0)
     printf("\n The given number is +VE ");
   else
     printf("\n The given number is -VE");

   return 0;
}
```

OUTPUT :

```
Enter any number : -3

The given number is -VE
```

```
Enter any number : 6

The given number is +VE
```

```
Enter any number : 0

The given number is Neutral
```

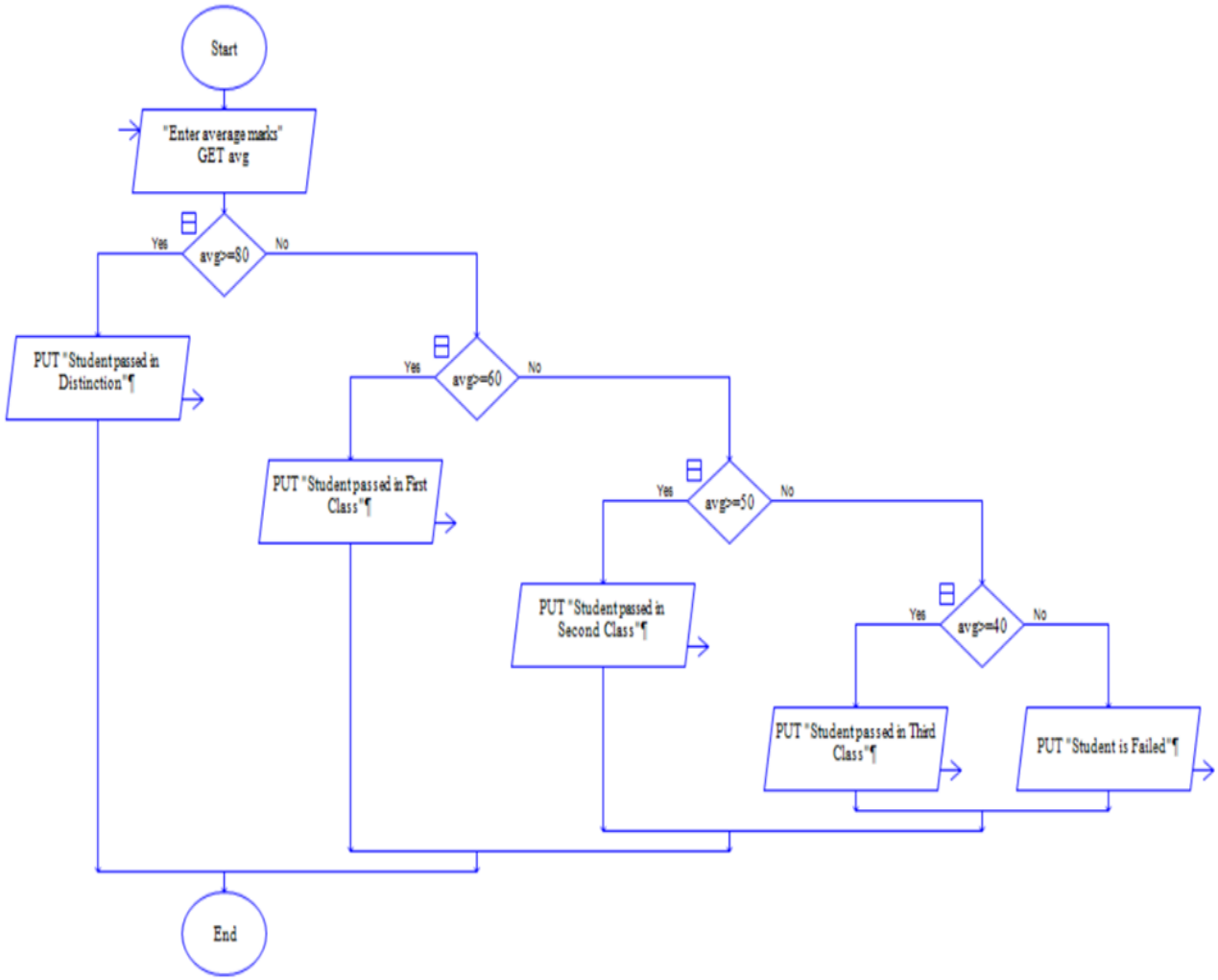
MODULE 4 – PROGRAM 4.5

AIM : write a C program to find the grade of a student using switch statement.

ALGORITHM :

STEP 1 : Start
STEP 2 : Read the avg marks of the student
STEP 3 : Check the (avg>=80) then display outstanding grade go to step 8
STEP 4 : Check the (avg<80)&(avg>=60) then display grade 1st class go to step 8
STEP 5 : Check (avg<60) & (avg>=50) then display grade 2nd class then go to step 8
STEP 6 : Check (avg<50) and (avg>=40) then display grade 3rd class then go to step 8
STEP 7 : Check (avg<40) then display grade fail
STEP 8 : Stop

FLOWCHART:



PROGRAM :

```
 #include<stdio.h>
void main()
 {
  int marks;
  printf("\n enter the average marks \t");
  scanf("%d",&marks);
  switch(marks/10)
  {
  case 0:
  case 1:
  case 2:
  case 3:   printf("\n fail \n");
                break;
  case 4:   printf("\n third class \n");
                break;

  case 5:  printf("\n second class \n");
                break;
  case 6: printf("\n first class \n");
                break;
  case 7:
  case 8:
  case 9:
  case 10 :     printf("\n distinction \n");
  } }
```

OUTPUT :

```
enter the average marks        30
 fail
enter the average marks        48
 third class

enter the average marks        57
 second class

enter the average marks        65
 first class

enter the average marks        80
 distinction
```

MODULE 4 – PROGRAM 4.6

AIM : Write a C Program to find the Roots of a quadratic equation.

FORMULA:

The roots of the quadratic equation: $x = (-b \pm \sqrt{D})/2a$, where $D = b^2 - 4ac$

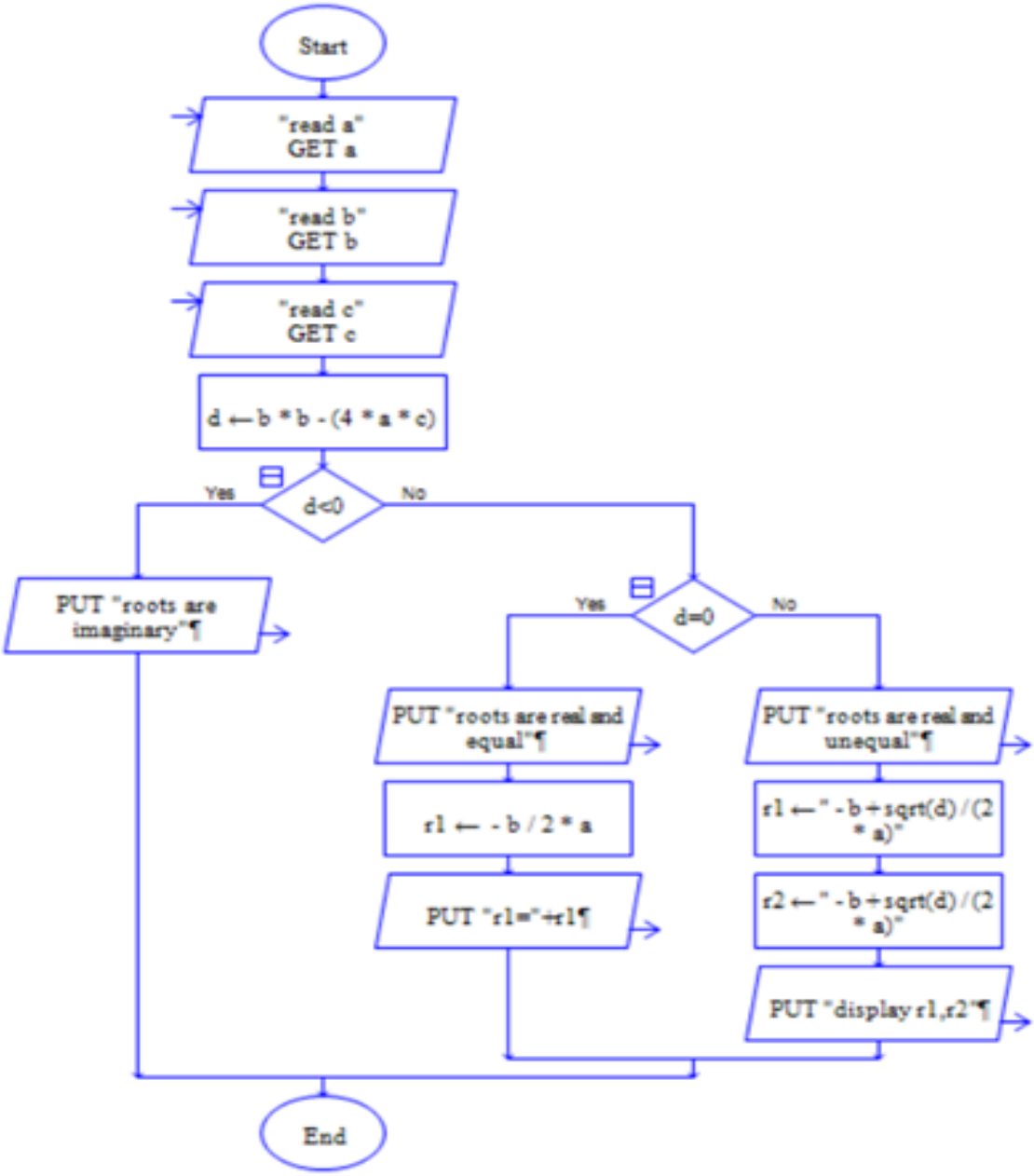
Nature of roots:

- D > 0, roots are real and distinct (unequal)
- D = 0, roots are real and equal (coincident)
- D < 0, roots are imaginary and unequal

ALGORITHM :

STEP-1 : ***Start***
STEP-2 : Read a, b , c values
STEP-3 : Calculate d=b*b-4*a*c
STEP-4 : If (d<0) Indicate roots are imaginary go to step 7
STEP-5 : If (d>0) indicate roots are real and unequal and then calculate
r1=-b+sqrt(d)/(2*a)
r2=-b-sqrt(d)/(2*a)
display r1, r2 and goto step 7
STEP-6 : Indicate roots are real and equal , calculate r=-b/(2a) and display r
STEP-7: stop

FLOWCHART:


Start
"read a"
GET a
"read b"
GET b
"read c"
GET c
d ← b * b - (4 * a * c)
Yes
d<0
No
PUT "roots are imaginary"¶
Yes
d=0
No
PUT "roots are real and equal"¶
r1 ← - b / 2 * a
PUT "r1="+r1¶
PUT "roots are real and unequal"¶
r1 ← " - b + sqrt(d) / (2 * a)"
r2 ← " - b + sqrt(d) / (2 * a)"
PUT "display r1,r2"¶
End

PROGRAM :

```
#include<stdio.h>
#include<math.h>
 void main()
 {
  int a,b,c,d;
  float r1,r2;
  clrscr();
  printf("\n enter values of a b c\t");
  scanf("%d %d %d",&a,&b,&c);
  d=b*b - 4*a*c;
  if(d==0)
  {
     printf("\n Roots are real & equal");
     r1=-b/(2.0*a);
     printf("\n root = %.2f",r1);
  }
  else if(d>0)
  {
     printf("\n Roots are real & unequal");
     r1=(-b+sqrt(d))/(2.0*a);
     r2=(-b-sqrt(d))/(2.0*a);
     printf("\n root1 = %.2f",r1);
     printf("\n root2 = %.2f",r2);
   }
  else
     printf("\n Roots are imaginary");
}
```

OUTPUT :

```
enter values of a b c  1 2 3
 Roots are imaginary

enter values of a b c  1 -5 6
 Roots are real & unequal
 root1 = 3.00
 root2 = 2.00

 enter values of a b c  1 -2 1
 Roots are real & equal
 root = 1.00
```

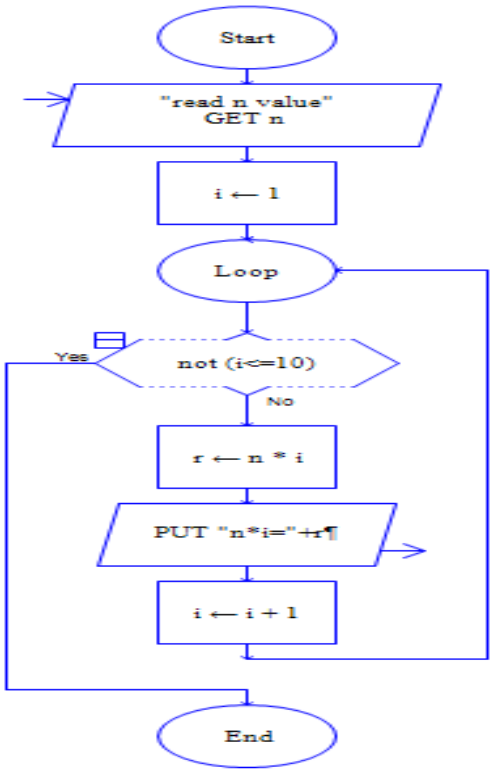
MODULE 4 – PROGRAM 4.7

AIM : write a C program write a program to print the multiplication table of the given numbers up to 10 terms.

ALGORITHM :

Step 1 : start
Step 2 : read n
Step 3 : i=1
Step 4: display (n*i)
Step 5 : i=i+1
Step 6 : repeat step 5,6 until i<=10
Step 7 : stop

FLOWCHART:



PROGRAM :

```
#include<stdio.h>
int main()
 {
  int n,f,i;
  printf("\n which table \t");
  scanf("%d",&n);
  printf(" \n multiplication table for %d is\n",n);
   for(i=1; i<=10; i++)
       printf("\n%d X %d = %d\n",i,n,i*n);
return 0;
}
```

OUTPUT :

```
which table    5

 multiplication table for 5 is

1 X 5 = 5
2 X 5 = 10
3 X 5 = 15
4 X 5 = 20
5 X 5 = 25
6 X 5 = 30
7 X 5 = 35
8 X 5 = 40
9 X 5 = 45
10 X 5 = 50
```

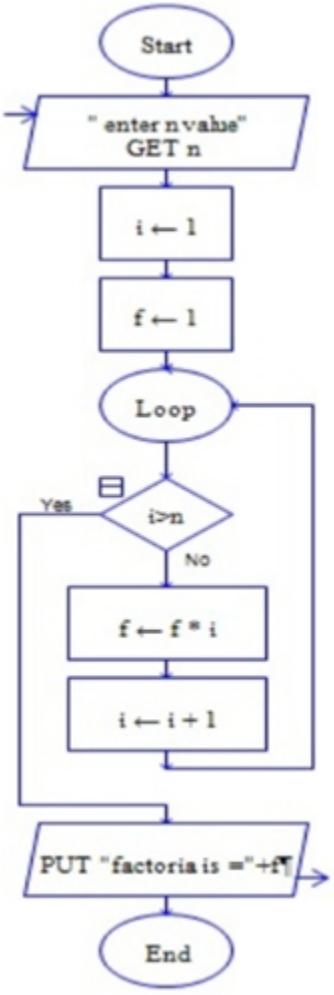
MODULE 4 - Program 4.8

AIM : Write a C program to Find the factorial of given number using any loop.

ALGORITHM :

STEP-1: Start
STEP-2: read n
STEP-3: f=1, i=1
STEP-4: repeat step 5 until i>n
STEP-5: calculate f=f*i , i=i+1
STEP-6: Display f
STEP-7: Stop

FLOWCHART:



PROGRAM :

```
#include<stdio.h>
int  main()
 {
  int n,i=1;
  long f=1;
  printf("\n enter the number \t");
  scanf("%d",&n);
  while(i<=n)
  {
    f=f*i;
    i++;
   }
     printf("\n factorial = %ld",f);
  return 0;
 }
```

OUTPUT :

```
 enter the number       5
 factorial = 120
```

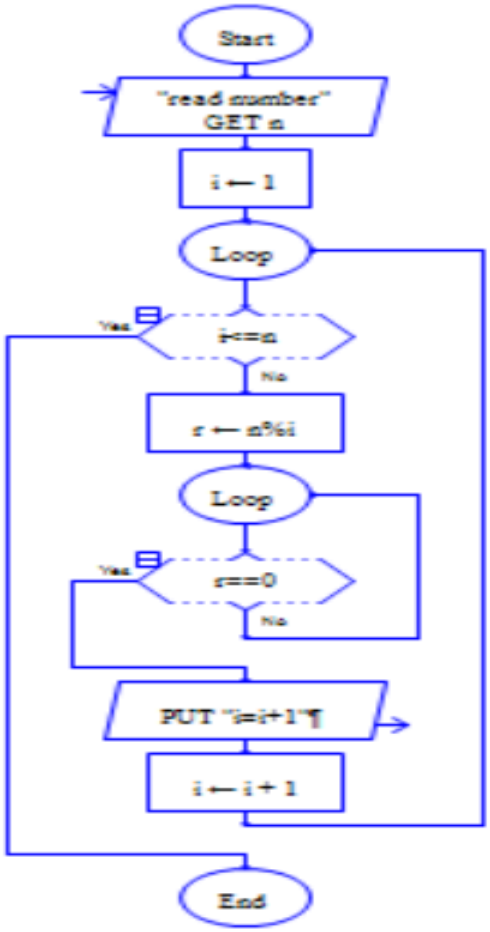
MODULE 4 – PROGRAM 4.9

AIM : Write a C Program to find the factors of a given a number.

ALGORITHM :

Step1 : start
Step2 : read n
Step3 : i=1
Step4 : repeat step 5 through 7 until while(i<=n/2)
Step5 : r=n%i
Step6 : if(r==0) then display i
Step7 : i=i+1
Step8 : stop

FLOWCHART:



PROGRAM :

```
#include<stdio.h>
int main()
 {
  int n,f,i;
  printf("\n enter the number \t");
  scanf("%d",&n);
  printf(" \n factors of %d are",n);
   for(i=1; i<=n/2; i++)
    if(n%i==0)
       printf("\t%d",i);
return 0;
}
```

OUTPUT :

```
enter the number      153
 factors of 153 are   1     3     9     17    51
```

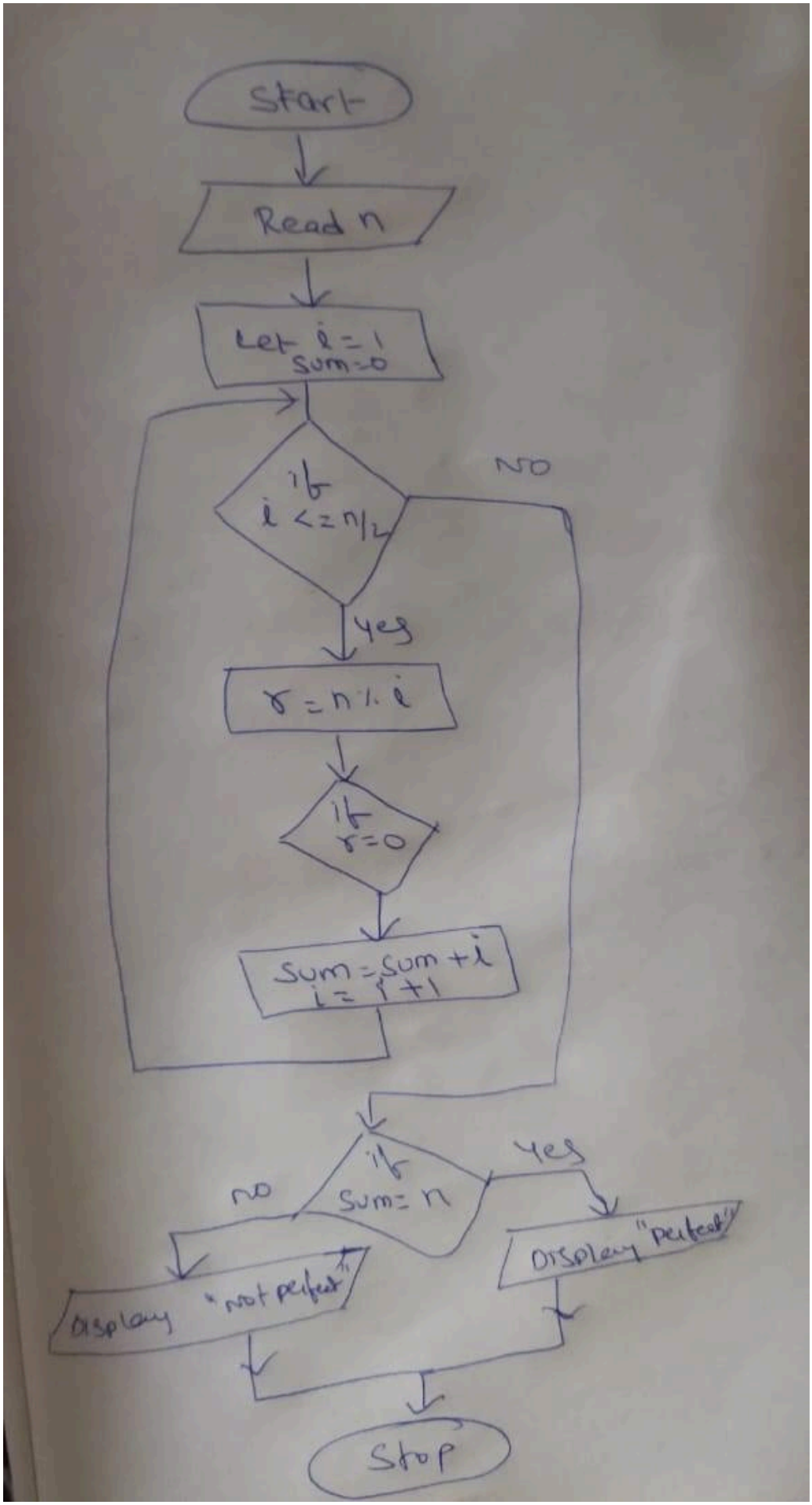
MODULE 4 – PROGRAM 4.10

AIM : Write a C Program to find the given number is perfect number or not.

ALGORITHM :

Step1 : start
Step2 : read n
Step3 : i=1
Step4 : repeat step 5 through 7 until while(i<=n/2)
Step5 : r=n%i
Step6 : if(r==0) then sum=sum+i
Step7 : i=i+1
Step8 : if(sum=n) then display " Perfect" otherwise display "Not Perfect"
Step9 : stop

FLOWCHART:


Start
Read n
Let i = 1
Sum = 0
if
i <= n/2
NO
yes
r = n % i
if
r = 0
Sum = Sum + i
i = i + 1
if
Sum = n
yes
no
Display "perfect"
Display "not perfect"
Stop

PROGRAM :

```
#include<stdio.h>
int main()
 {
  int n,f,i,sum=0;
  printf("\n enter the number \t");
  scanf("%d",&n);
  for(i=1; i<=n/2; i++)
    if(n%i==0)
       sum=sum+i;
  if(sum==n)
    printf(" %d is perfect number ",n);
  else
    printf(" %d is not a perfect number ",n);
return 0;
}
```

OUTPUT :

```
enter the number        6
6 is perfect number
```

```
enter the number        121
121 is not a perfect number
```