Date - 16/08/19

(01) Write a c Program to find out the value of F(2.55) using Newton's Forward Interpolation Formula from the following table :

x :	2.00	2.25	2.50	2.75	3.00
F(x) :	9.00	10.06	11.25	12.56	14.00

Algorithm for Newton's Forward Interpolation formula :-

```
Step 1 :  Start
Step 2 :  Take range n
Step 3 :  read $x_i, y_i$ for $i = 1$ to $n$
Step 4 :  read $x_1$
Step 5 :  Set $p = n-1$, $s = y[0]$, $j = 0$, $s_1 = 0$
Step 6 :  take $u = (x_1 - x[0]) / (x[1] - x[0])$
Step 7 :  for $i = 1$ to $n-1$
            for $j = 0$ to $p$
              $y[j] = y[j+1] - y[j]$
              $S_1 = S_1 + (u \times c \times y[0]) / fact(i)$
Step 8 :  $c = c \times (u - i)$
          $p = p - 1$
Step 9 :  Set $S = S1 + S$
Step 10 : Print S
Stop 11 : Stop
```

C-Code for Newton's Forward Interpulation

```
# include <stdio.h>
  int fact (int x)
  {
   int i, f, f=1;
   for (i=1; i<=x; i++)
       f = f*i;
       return (f);
  }
  void main () {
   float x[10], y[10], x1, u, h, p, s, s1=0, c=1;
    int n, i, j;
   Printf("Enter number of terms:");
   Scanf("%d", &n);
   Printf("Enter the values of x and y \n");
  for (i=0; i<n; i++)
   Scanf("%f %f", &x[i], &y[i]);
   Printf(" Enter the value of x to find :");
    Scanf("%f", &x1);
     P= n-1; S = y[0];
     u = (x1 - x[0])/x[1]-x[0]);
  for( i=1; i<n-1; i++)
      {
      for (j=0; j<p; j++)
    s[j] = y[j+1] - y[j];
    s1 = s1 + (u * c * y[0]) / fact (i);
    c = c * (u-i);
    p = p-1;
  }
  s = s1 + s;
  printf(" \n Ans is = %f", s);
}
```

⊛ **Output:-**

```
Enter number of terms = 5
Enter values of x and y.
 2.00
 9.00
 2.25
 10.06
 2.50
 11.25
 2.75
 12.56
 3.00
 14.00
Enter the values of x to find : 2.35

Ans is = 10.521408.
```

Date- 30/08

2. Write a C program to find out the value of F (4.5) using Newton's Backword Interpolation. Formula from the following table :-

x :	2	3	4	5
f(x) :	11	15	23	39

Algorithm for Newton's Baackward Interpolation formula :-

```
Step 1 : start
Step 2 : Take range of n
Step 3 : read xi, yi for i=1 to n
Step 4 : read xp, j=1
Step 5 : While (j<n)
             for (i=0 to n-j)
             ~~Print f~~ Print y[i]
Step 6 : set p=n-1, s=y[n-1], s, s1=0, c=1
Step 7 : take u= (x1 - x[n-1])/(x[1]-x[0]);
Step 8 : for i=1 to n-1
             s1 = s1 + (u*c*4[p-1])/fact(i)
Step 9 : take c= c*(u+i)
         take p= p = p-1
Step 10 : set s= s1+s
Step 11 : Print s
Step 12 : stop
```

C - Code for Newton's Backward Interpolation :-

```
# include <stdio.h>
   int fact (int x) {
          int i,f, f= 1;
        for (i=1 ; i<= x; i++)
          f=f* i;
          return (f);
     void main ()
      { float x[10], y[10], x1, v, h, S, S1 = 0, c=1;
       int i, n, j=1 , , P;
     Printf(" Enter how many terms : ");
      scanf(" %d", &n);
     Printf(" Enter values for x \n");
      for( i=0; i<n ; i++)
      scanf(" %f ; &x[i]));
     Printf(" Enter values for y \n);
      for ( i=0; i<n; i++)
      Scanf(" %f", &y[i]);
     Printf(" Enter values of to find : ");
      Scanf(" %f", &x1);
     While (j<n) Printf(" \n Difference table is \n");
     { for (i=0 ; i< n-j ; i++)
          { ~~for(j=0;~~ y[i] = y[i+1] - y[i] ;
printf("%f \t", y[i]);
} printf(" \n");
  j++;
}
} p = n-1 ; s= y[n-1] ;
u = (x1 - x[n-1]) / (x[1] - x[0]);
for (i=1 ; i<n-1; i++)
{ s1 = s1+(u*c*y[p-1])/fact(i);
   c = c*(u+i);
   p= p-1;
} s= s1+s ;
printf(" \n Ans is= %f ",s);
```

Output :-

```
Enter how many terms : 4
Enter values of x     : 2  3  4  5
Enter values of y     : 11 15 23 39
Enter value of x to find : 4.5
Difference table is
   4.000000   8.000000
   4.000000              16.000000
              8000000
   4.000000

Ans is = 29.750000.
```

Date – 30/08/19

③ Write a C Program to find out the value of f(10) using Lagrange's Interpolation formula. From the following table :

x :	5	6	9	11
f(x) :	12	13	14	16

• Algorithm for Lagrange's Interpolation formula :–

```
Step 1 : Start
Step 2 : Take range of n
Step 3 : Read $x_i$ & $y_i$ for i=1 to n
Step 4 : Read xp
Step 5 : Initialize y=0, S[i], N, D
Step 6 : for i = 0 to n
            N = 1 , D = 1
          for i = 0 to n
            if $j \neq i$
              N = N * (x - x[i])
              D = D * [x[i] - x[j])
 Step 7 : S[j] = N/D
 Step 8 : y = y + S[j] * y[j]
 Step 9 : Print y
 Step 10 : Stop
```

• C- code for Langrange's Interpolation :-

```
#Include <stdio.h>
  main ()
  { int i, j=1, n;
   float x[10], y[10], s[10], x, y=0, N, D;
  Printf(" Enter number of terms : ");
   scanf("%d", &n);
  Printf("Enter values for x\n");
  for( i=0 ; j<n ; i++)
   Scanf("%f", &x[i]);
   Printf(" Enter values for y\n");
   for( i=0; i<n ; i++)
    Scanf("%f", &y[i]);
    Printf(" Enter values of x for find : ");
    Scanf("%f", &x);
    for( j=0; j<n ; j++)
     { N=1
       D=1
      for (i=0;j<n ; i++)
     { if(j != i)
      { N = N*(x-x[i]);
        D=D*(x[i]-x[j]);
      }
     }
    s[i] = N/D;
    y = y + s[i] *y[i];
  }
  printf(" Ans is = %f", y);
}
```

Output :-

```
Enter number of terms : 4
Enter values of x
    5
    6
    9
    11
Enter values of y
    12
    13
    14
    16

Enter value of x for find : 10

Ans is = 14.666666.
```

20/9/19

04 Write a C Program to Trapizoidal Method.

⇒ **Algorithm :-**

```
Step 1 : start the Program.
Step 2 : define the fn $f(x)$
Step 3 : read $a, b, n, S=0, t$.
Step 4 : Set $x_0 = a$ . and $x_n = b$ .
Step 5 : take $h = \frac{b-a}{n}$
Step 6 : For $i = 0(1)n$
Step 7 :     $x_i = x_0 + ih$ , $y_i = f(x_i)$
Step 8 : for $i = 1(1)n-1$
Step 9 :     $S = S + f(x_i)$
Step 10 : $t = \left(\frac{h}{2}\right) * (y_0 + y_n + 2S)$
Step 11 : Print the value of $x$ .
Step 12 : Stop the Program.
```

C Programm :-

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
float f(float x)
{ return (1/(1+pow(x,2)));
}
```

```
void main ( )
 { int i, n;
  float x0, xn, h, y[20], s0, se, ans, x[20];
  Printf("\n Enter values of x0 ,xn, h :\n");
  Scanf("%f%f%f", &x0, &xn ,&h);
   n = (xn-x0)/h ;
   if(n%2==1)
    { n=n+1;
    }
    h= (xn-x0)/n;
  Printf("\n ~~re~~fined value of n and h are : %d%f\n",
        n,h);
   Printf("\n y values\n");
   for(i=0; i<=n; i++)
    { x[i] = x0+ i*h;
    y[i] = f(x[i]);
   Printf("\n%f\n" ,y[i]);

   } s0 = 0;
     se = 0;
   for (i=1; i<n; i++)
    { if(i%2==1)
       {
```

```
    so = so+y[i];
    }
    else
    {
     se = se+y[i];
     }
   }
  ans = h/3* (y[0]+y[n]+4*so+2*se);
  printf("\n final integration is %f", ans);
  getch();
  }
```

Output

```
  Enter values of x0, xn ,h:  0   3   0.5
  refined  value of n and h are:  6   0.500000
  Y values
  1.000000
  0.800000
  0.500000
  0.307692
  0.200000
  0.137931
  0.100000
final integration is 1.247082.
```

Scanned by CamScanner

5. Write c Program using simpson's $\frac{1}{3}$ Rule

→ Algorithm

```
Step 1 :   Start;
Step 2 :   Input function f(x);
Step 3 :   Read a, b, n ;
Step 4 :   Compute h = (b-a)/n ;
Step 5 :   Sum = [f(a) + f(a+nh)];
Step 6 :   for i = 1 to n-1 step 2 do
               Compute sum = sum + 4*f(a+ih) + 2*f(a+
                       (i+1)h);
           end for;
Step 7 :   Compute result = sum * h/3;
Step 8 :   Print result;
Step 9 :   Stop.
```

Program :

```
# include <stdio.h>
# include <math.h>
float f(float x)
        {
        return exp(x);
        }
int main()
    {
```

```
float a,b,n,h, sum1 =0; sum2=0, sum , yo,yn;
int i;
Printf(" Enter the upper limit : ");
 Scanf(" %f", &b);
Printf(" Enter the lower limit : ");
 Scanf(" %f", &a);
Printf(" Enter the number of intervals : ");
Scanf(" %f", &n);
  h = (b-a)/n;
  yo = f(a+0*h);
  yn = f(a+n*h);
  for( i=1; i<n; i++)
     if( i%2 = = 0)
       sum1= sum1+ f(a+i*h);

     else
       sum2 = sum2 + f(a+i*h);
  sum = (h/3) *(yo+ yn + 2*sum1+ 4* sum2);
  Printf(" Answer : %.2f", sum);
   getch();
   return 0;

  }
```

Output:

```
Enter the upper limit : 0.6
Enter the lower limit : 0
Enter the number of intervals : 10
Answer : 0.82.
```

6. Write a C Program using Bisection Method.

Algorithm:

```
Step 1 :  Start the Program ;
Step 2 :  define the function f(x);
Step 3 :  select the interval a, b in which the
          root lies where f(a). f(b) <0 ;
Step 4 :  Calculate $x = \frac{a+b}{2}$ ;
Step 5 :  if f(b) f(x) <0 :
              set a = x ;
              otherwise set b = x ;
Step 6 :  if |a-b| < ε ; ε being the Prescribed
          accuracy then goto step 7;
~~Step~~ -  else goto step 4 ;
Step 7 :  Print the value of x;
Step 8 :  Stop the Program;
```

C Program:

```
  #include <stdio.h>
 # include <conio.h>
 # include <math.h>
 float F (float x)
  { return (x*x*x-(5*x)+1);
  }
```

```
void main ()
{ int i=0;
 float a,b,c,err,temp;
  clrscr ();
   Printf(" enten permissive error \n");
   Scanf("%f",&err);
    do
    {Printf(" enten value of a & b \n");
     scanf("%f%f",&a,&b);
    }
    while( F(a)*F(b)>0);
     c=0;
     do
     {temp = c;
       c= (a+b) /2;
       if(F(a)*f(b)<0)
       {
        b=c;
       }
       else
       {a=c;
       }
       i++;
    Printf(" Iteration %d", i);
printf(" %f\n", c);
} while (fabc (temp-c)> err);
 printf(" root of equation is %f ",c);
 getch();
}
```

output :

```
enter permissive error : 0.0005
enter value of a & b : 2   3

iteration 1 : 2.500000
iteration 2 : 2.250000
iteration 3 : 2.125000
iteration 4 : 2.062500
iteration 5 : 2.093750
iteration 6 : 2.1093750
iteration 7 : 2.117188
iteration 8 : 2.121094
iteration 9 : 2.123047
iteration 10 : 2.124023
iteration 11 : 2.124512
root of equation is 2.124512.
```