CENTRAL CALCUTTA POLYTECHNIC

21, Convent Road, Philips, Sealdah, Kolkata, West Bengal 700014

DEPT. : COMPUTER SCIENCE AND TECHNOLOGY

- NAME : SUMAN MONDAL
- ROLL : DCCPCSTS6
- NUMBER : 10005537
- REG NUMBER : D192005242
- SUBJECT : NUMERICAL METHODS
- SESSION : 2021 - 2022
- EMAIL : SUMAN.MONDAL@OUTLOOK.IN

Contents

Chapter 1

Numerical Methods Assignments

1.0.1 Write a C Program to find out the value of f(2.35) 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

Source Code :

```
#include <math.h>
#include <stdio.h>
int main()
{
        float x[10], y[10][10], h, u1, u, fx, fy, fact;
        int i, j, n, ch = 30;
        printf("How many terms you want to enter : ");
        scanf("%d", &n);
        printf("Enter the value of X and Y (X,Y): ");
        for (i = 0; i < n; i++)
                scanf("%f,%f", &x[i], &y[i]);
        for (j = 1; j < n; j++)
        {
                for (i = 0; i < n - j; i++)
                {
                        y[i][j] = y[i + 1][j - 1] - y[i][j - 1];
                }
        }
        printf("_____\n");
        printf("\n  x\t    y\t    y1\t    y2\t    y3\t    y4");
        printf("\n_____\n");
        for (i = 0; i < n; i++)
        {
                printf("%.2f", x[i]);
                j = 0;
                while (j < n - i)
                {
                        printf("        %.3f", y[i][j]);
                        j++;
                }
                printf("\n");
        }
        printf("\nEnter the value of x for which you wants to find Y : ");
        scanf("%f", &fx);
```

```
        h = x[1] - x[0];
        u = (fx - x[0]) / h;
        fy = y[0][0];
        u1 = u;
        fact = 1;
        for (i = 1; i < n; i++)
        {
                fy = fy + (u1 * y[0][i]) / fact;
                u1 = u1 * (u1 - i);
                fact = fact * (i + 1);
        }
        printf("\nY(%f)=%.3f\n", fx, fy);
}
```

Program Output :

```
How many terms you want to enter : 5
Enter the value of X and Y (X,Y): 2.00,9.00
2.25,10.06
2.50,11.25
2.75,12.56
3.00,14.00
_____________________________________________________

  x         y       y1      y2      y3      y4
_____________________________________________________
2.00    9.000   1.060   0.130   -0.010  0.020
2.25    10.060  1.190   0.120   0.010
2.50    11.250  1.310   0.130
2.75    12.560  1.440
3.00    14.000

Enter the value of x for which you wants to find Y : 2.35

Y(2.350000)=10.524
```

1.0.2 Write a C Program to find out the value of f(4.25) using Newton's Backward Interpolation Formula from the following table

x:	2.5	3.0	3.5	4.0	4.5
f(x):	9.75	12.75	15.70	19.52	23.75

Source Code :

```
#include <stdio.h>
int fact(int);
void main()
{
        int n, i, j, ch = 30;
        float arr[10][11], px = 1, x, y, p, h;
        printf("\nHow many terms you want to enter:");
        scanf("%d", &n);
        printf("\nEnter the value of X and Y:");
        for (i = 0; i < n; i++)
        {
                printf(" X%d=", i + 1);
                scanf("%f", &arr[i][0]);
                printf(" Y%d=", i + 1);
                scanf("%f", &arr[i][1]);
```

```
        }
        for (j = 2; j <= n; j++)
        {
                for (i = 0; i < n - 1; i++)
                        arr[i][j] = arr[i + 1][j - 1] - arr[i][j - 1];
        }
        printf("_____________________________________________________");
        printf("\n\t  x\t    y\t    y1\t    y2\t    y3\t    y4");
        printf("\n_____________________________________________________");
        for (i = 0; i < n; i++)
        {
                printf("\n");
                for (j = 0; j < n + 1 - i; j++)
                        printf("\t%.4f", arr[i][j]);
        }
        printf("\nEnter the value of x for f(x): ");
        scanf("%f", &x);
        h = arr[n - 1][0] - arr[n - 2][0];
        p = (x - arr[n - 1][0]) / h;
        y = arr[n - 1][1];
        for (i = 1; i < n; i++)
        {
                px = px * (p + (i - 1));
                y = y + (arr[n - 1 - i][i + 1] * px) / fact(i);
        }
        printf("\nthe value of f(x) at x=%f is %f", x, y);
}

int fact(int n)
{
        int f = 1, i;
        for (i = 1; i <= n; i++)
                f = f * i;
        return (f);
}
```

Program Output :

```
→ gcc Q02.c && ./a.out

How many terms you want to enter:5

Enter the value of X and Y: X1=2.5
 Y1=9.75
 X2=3.0
 Y2=12.75
 X3=3.5
 Y3=15.70
 X4=4.0
 Y4=19.52
 X5=4.5
 Y5=23.75

_____________________________________________________
          x         y       y1      y2      y3      y4
_____________________________________________________
        2.5000  9.7500  3.0000  -0.0500 0.9200  -1.3800
        3.0000  12.7500 2.9500  0.8700  -0.4600
        3.5000  15.7000 3.8200  0.4100
        4.0000  19.5200 4.2300
        4.5000  23.7500
Enter the value of x for f(x): 4.25

the value of f(x) at x=4.250000 is 21.666405%
```

1.0.3 Write a C Program to find out the value of f(4.25) using Newton's Divide Difference Interpolation Formula from the following table

x:	2.5	3.0	4.5	4.75	6.0
f(x):	8.85	11.45	20.66	22.85	38.60

Source Code :

```c
#include <math.h>
#include <stdio.h>
int main()
{
        float x[10], y[10][10], sum, p, u, temp;
        int i, n, j, k = 0, f, m;
        float fact(int);

        printf("\nHow many terms you want to enter: ");
        scanf("%d", &n);
        printf("\nEnter the value of X and Y:");
        for (i = 0; i < n; i++)
        {
                printf(" X%d=", i + 1);
                scanf("%f", &x[i]);
                printf(" Y%d=", i + 1);
                scanf("%f", &y[k][i]);
        }

        for (i = 1; i < n; i++)
        {
                k = i;
                for (j = 0; j < n - i; j++)
                {
```

```
                        y[i][j] = (y[i - 1][j + 1] - y[i - 1][j]) / (x[k] - x[j]);
                        k++;
                }
        }
        printf("_____\n");
        printf("\n  x\t   y\t    y1\t    y2\t    y3\t    y4");
        printf("\n_____\n");
        for (i = 0; i < n; i++)
        {
                printf("\n %.3f", x[i]);
                for (j = 0; j < n - i; j++)
                {
                        printf("    ");
                        printf(" %.3f", y[j][i]);
                }
        }

        printf("\n\nEnter the value X for f(x): ");
        scanf("%f", &p);

        i = 0;
        do
        {
                if (x[i] < p && p < x[i + 1])
                        k = 1;
                else
                        i++;
        } while (k != 1);
        f = i;

        sum = 0;
        for (i = 0; i < n - 1; i++)
        {
                k = f;
                temp = 1;
                for (j = 0; j < i; j++)
                {
                        temp = temp * (p - x[k]);
                        k++;
                }
                sum = sum + temp * (y[i][f]);
        }
        printf("\n\n f(%.2f) = %f ", p, sum);
}
```

Program Output :

```
→ gcc Q03.c && ./a.out

How many terms you want to enter: 5

Enter the value of X and Y: X1=2.5
 Y1=8.85
 X2=3.0
 Y2=11.45
 X3=4.5
 Y3=20.66
 X4=4.75
 Y4=22.85
 X5=6.0
 Y5=38.60

-------------------------------------------------------------

  x         y         y1       y2       y3       y4

-------------------------------------------------------------

 2.500    8.850    5.200    0.470    0.457    -0.029
 3.000    11.450    6.140     1.497    0.354
 4.500    20.660    8.760     2.560
 4.750    22.850    12.600
 6.000    38.600

Enter the value X for f(x): 4.25


 f(4.25) = 18.712500
```

1.0.4 Write a C Program to evaluate $\int_2^1 \frac{1}{1+x^2}\,dx$ using Trapezoidal rule with 6 intervals

Source Code :

```c
#include <stdio.h>
#include <math.h>

double f(double x)
{
        return 1 / (1 + pow(x, 2));
}

int main()
{
        int n, i;
        double a, b, h, x, sum = 0, integral;

        printf("\nEnter the no. of sub-intervals: ");
        scanf("%d", &n);
        printf("\nEnter the initial limit: ");
        scanf("%lf", &a);
        printf("\nEnter the final limit: ");
        scanf("%lf", &b);

        h = fabs(b - a) / n;
        for (i = 1; i < n; i++)
        {
                x = a + i * h;
```

```
                sum = sum + f(x);
        }
        integral = (h / 2) * (f(a) + f(b) + 2 * sum);

        printf("\nThe integral is: %lf\n", integral);
        return 0;
}
```

Program Output :

```
Enter the no. of sub-intervals: 6

Enter the initial limit: 2

Enter the final limit: 1

The integral is: 0.175462
```

1.0.5 Write a C Program to evaluate $\int_2^1 \frac{x}{1+x}\,dx$ using Simpson's 1/3rd Rule with 6 intervals

Source Code :

```
#include <math.h>
#include <stdio.h>

double f(double x)
{
        return (x / (1 + x));
}

int main()
{
        int n, i;
        double a, b, h, x, sum = 0, integral;

        printf("\nEnter the no. of sub-intervals(EVEN): ");
        scanf("%d", &n);
        printf("\nEnter the initial limit: ");
        scanf("%lf", &a);
        printf("\nEnter the final limit: ");
        scanf("%lf", &b);

        h = fabs(b - a) / n;
        for (i = 1; i < n; i++)
        {
                x = a + i * h;
                if (i % 2 == 0)
                        sum = sum + 2 * f(x);
                else
                        sum = sum + 4 * f(x);
```

```
        }
        integral = (h / 3) * (f(a) + f(b) + sum);

        printf("\nThe integral is: %lf\n", integral);
}
```

Program Output :

```
Enter the no. of sub-intervals(EVEN): 6

Enter the initial limit: 2

Enter the final limit: 1

The integral is: 0.698429
```

1.0.6 Write a C Program to find the root of the equation $x^3 + x^2 + x + 7 = 0$ using Bisection Method

Source Code :

```
#include<stdio.h>
#include<math.h>

#define f(x) pow(x,3)+pow(x,2)+x+7

int main()
{
        float x0, x1, x2, f0, f1, f2, e;
        int step = 1;

        up:
        printf("\nEnter two initial guesses:\n");
        scanf("%f%f", &x0, &x1);
        printf("Enter tolerable error:\n");
        scanf("%f", &e);

        f0 = f(x0);
        f1 = f(x1);

        if( f0 * f1 > 0.0)
        {
                printf("Incorrect Initial Guesses.\n");
                goto up;
        }
    /* Implementing Newton Bisection Method */
        printf("\nStep\t\tx0\t\tx1\t\tx2\t\tf(x2)\n");
        do
        {
                x2 = (x0 + x1)/2;
                f2 = f(x2);

                printf("%d\t\t%f\t%f\t%f\t%f\n",step, x0, x1, x2, f2);
```

```
                if( f0 * f2 < 0)
                {
                        x1 = x2;
                        f1 = f2;
                }
                else
                {
                        x0 = x2;
                        f0 = f2;
                }
                step = step + 1;
        }while(fabs(f2)>e);
        printf("\nRoot is: %f", x2);
  printf("\n");
  return 0;
}
```

Program Output :

```
Enter two initial guesses:
-2
-3
Enter tolerable error:
0.0001

Step            x0              x1              x2              f(x2)
1               -2.000000       -3.000000       -2.500000       -4.875000
2               -2.000000       -2.500000       -2.250000       -1.578125
3               -2.000000       -2.250000       -2.125000       -0.205078
4               -2.000000       -2.125000       -2.062500       0.417725
5               -2.062500       -2.125000       -2.093750       0.111481
6               -2.093750       -2.125000       -2.109375       -0.045498
7               -2.093750       -2.109375       -2.101562       0.033315
8               -2.101562       -2.109375       -2.105469       -0.006010
9               -2.101562       -2.105469       -2.103516       0.013673
10              -2.103516       -2.105469       -2.104492       0.003836
11              -2.104492       -2.105469       -2.104980       -0.001086
12              -2.104492       -2.104980       -2.104736       0.001376
13              -2.104736       -2.104980       -2.104858       0.000145
14              -2.104858       -2.104980       -2.104919       -0.000470
15              -2.104858       -2.104919       -2.104889       -0.000163
16              -2.104858       -2.104889       -2.104874       -0.000009

Root is: -2.104874
```

1.0.7 Write a C Program to find the root of the equation $x^3 - x - 3 = 0$ using Newton Raphson Mehtod

Source Code :

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>

#define f(x) pow(x,3) - x - 3
#define g(x) 3*pow(x,2) - 1

int main()
{
        float x0, x1, f0, f1, g0, e;
        int step = 1, N;

        printf("\nEnter initial guess: ");
```

```
        scanf("%f", &x0);
        printf("Enter tolerable error: ");
        scanf("%f", &e);
        printf("Enter maximum iteration: ");
        scanf("%d", &N);
        /* Implementing Newton Raphson Method */
        printf("\nStep\t\tx0\t\tf(x0)\t\tx1\t\tf(x1)\n");
        do
        {
                g0 = g(x0);
                f0 = f(x0);
                if(g0 == 0.0)
                {
                        printf("Mathematical Error.");
                        exit(0);
                }


                x1 = x0 - f0/g0;


                printf("%d\t\t%f\t%f\t%f\t%f\n",step,x0,f0,x1,f1);
                x0 = x1;

                step = step+1;

                if(step > N)
                {
                        printf("Not Convergent.");
                        exit(0);
                }

                f1 = f(x1);

        }while(fabs(f1)>e);

        printf("\nRoot is: %f", x1);
  printf ("\n");
  return 0;
}
```

Program Output :

```
Enter initial guess: 2
Enter tolerable error: 0.00001
Enter maximum iteration: 10

Step            x0              f(x0)           x1              f(x1)
1               2.000000        3.000000        1.727273        0.000000
2               1.727273        0.425996        1.673691        0.425996
3               1.673691        0.014723        1.671703        0.014723
4               1.671703        0.000020        1.671700        0.000020

Root is: 1.671700
```