```
/*
Name: Divyashree HB
Program 2b:
	The following program computes the coefficients of an interpolating polynomial
using newtons forward differance method.
Furthermore, the program computes the first and secound derivative using Three-point
midpoint formula.
	This program runs on the function f(x)=((e)^x)^2. Moreover, this program also
prompts user the choice of inputing f(x) values or compute itself.
Program calculates:
	Datapoints(equally interval points)
	f(x) values using f(x)=((e)^x)^2 for all datapoints
	Computes the newtons forward differance table (saved in funval(functional_values))
	Computes the first and secound derivative using Three-point midpoint formula.
User Input:
	Intervals
	Number od datapoints
	A choice to input f(x) or functional_values through console.
	X value of the interpolating polynomial.
Output:
	Prints the table.
	f'(x) and f``(x)
*/


#include<iostream>
#include<vector>
#include <math.h>//exp
#include<iomanip>//precission

using namespace std;

void printheading() {
	cout << "Divyashree H B" << endl;
	cout << "Newtons differential differance" << endl;
}
double fun_x(double x) {
	cout << fixed;
	std::setprecision(6);
	return exp(x*x);
}
vector<double> computeXval(int x, int y, int points) {
	vector<double> fun(points);
	//fun = computeXval(x, y, points);
	cout << fixed;
	std::setprecision(6);
	double tempsu = (y - x) / (double)(points - 1);
	fun[0] = (x);
	for (int i = 1; i < points; i++)
	{
		//cout << tempsu << endl;
		fun[i] = fun[i - 1] + tempsu;
		//cout << fun[i] << endl;
	}
	return fun;
	//cout <<"datapoint"<< fun[0] << endl;
}
```

```
vector<vector< double>> computefx(vector<vector< double>> &funval, vector<double> fun,
char testf, int datapt) {
	if (testf == 'N' || testf == 'n')
	{
		cout << endl << "Enter the values :" << endl;
		double tempin;
		for (int i = 0; i < datapt; i++)
		{
			cin >> tempin;
			funval[i][0] = (tempin);
		}
	}
	else
	{
		double temp;
		for (int i = 0; i < datapt; i++)
		{
			temp = fun_x(fun[i]);
			//cout << "y0\t" << temp<<endl;
			funval[i][0] = temp;
		}

	}
	return funval;
}
vector<vector< double>> filltable(vector<vector< double>> &funval, vector<double> fun,
int datapt) {
	for (int i = 1; i < datapt; i++) {
		for (int j = 0; j < (datapt - i); j++) {
			//cout << i << " " << j << " " << funval[i - 1][j + 1] << " " <<
funval[i - 1][j] << " " << endl;
			funval[j][i] = ((funval[j + 1][i - 1]) - (funval[j][i - 1])) /
(fun[j + i] - fun[j]);
		}//for (int i = 0; i < tempfun.size(); i++)cout << tempfun[i] << "tempfun"
<< endl;

	}
	return funval;
}
void printtable(vector<vector< double>> funval, vector<double> fun, int datapt) {
	cout << endl << "Differantial Table :" << endl;
	//// printing the elements
	for (int i = 0; i < datapt; i++)
	{
		cout << fun[i] << "  ";
		for (int j = 0; j < (datapt - i); j++) {
			cout << funval[i][j] << "  ";
		}
		cout << endl;
	}
}
void computepx(vector<vector< double>>funval, vector<double> fun) {
	double X;
	cout << "\n" << "Enter the value of X:" << endl;
	cin >> X;
	//calculating P(X) for a given X
	//double prod_x=1;//(x-xi)
	double sumval = funval[0][funval[0].size() - 1];//p(x)
```

```
        for (int i = funval[0].size() - 2; i >= 0; i--)
        {
                sumval = funval[0][i] + (X - fun[i])*sumval;
        }

        cout << "P(" << X << ")    =  " << sumval << endl;
        //condition to quiet
        cout << endl;
}

int main() {

        double Xo;
        double h, f1x, f2x;
        printheading();
        int mov = 1;//exit
        double x, y;//intervals
        char testf;//testcase for N and Y
        int datapt;
        while (mov == 1)
        {
                //read data
                cout << endl << "Enter N to give functional values else enter Y" << endl;
                cin >> testf;
                cout << endl << "Enter the intervals " << endl;
                cin >> x >> y;
                //cout << x << y<<endl;
                cout << endl << "Enter the number of data points" << endl;
                cin >> datapt;
                vector<double> fun = computeXval(x, y, datapt);
                //calculate data point values and save in fun array i.e  x value array
                vector<vector< double>> funval(datapt, vector<double>(datapt));      //f(x)
value array
                int temp = 1;
                while (temp == 1) {
                        int chint;
                        cout << "\n" << "Enter 1 to print the difference table \n      2 To
compute interpolating polynomial \n      3 For computing first derivative \n      ";
                        cout << "4 For computing seccond derivative \n      5 For computing
both derivative \n      Anything else to break :" << endl;
                        cin >> chint;
                        switch (chint) {
                        case 1:
                                funval = computefx(funval, fun, testf, datapt);
                                // fun vector holds all data points in the given interval
                                //funval[i][] in i all the initial f(x) or functional values
are found
                                //starting with iteration
                                funval = filltable(funval, fun, datapt);
                                printtable(funval, fun, datapt);
                                //computepx(funval, fun);
                                break;
                        case 2:
                                funval = computefx(funval, fun, testf, datapt);
                                // fun vector holds all data points in the given interval
                                //funval[i][] in i all the initial f(x) or functional values
are found
                                //starting with iteration
```

```
                    funval = filltable(funval, fun, datapt);
                    printtable(funval, fun, datapt);
                    computepx(funval, fun);
                    break;
                case 3:
                    cout << "\n" << "Enter the value of Xo in f`(Xo) :" << endl;
                    cin >> Xo;
                    cout << "The following approximation is calculated using 
three-point midpoint formula." << endl;
                    h = (y - x) / (double)(datapt - 1);
                    //for (int i = 1; i < datapt - 2; i++){}
                    f1x = (1 / (2 * h))*(fun_x(Xo + h) - fun_x(Xo - h));
                    cout << "f`(" << Xo << ")   =  " << f1x << endl;
                    break;
                case 4:
                    cout << "\n" << "Enter the value of Xo in f``(Xo) :" << endl;
                    cin >> Xo;
                    cout << "The following approximation is calculated using 
three-point midpoint formula." << endl;
                    h = (y - x) / (double)(datapt - 1);
                    f2x = (1 / (h * h))*(fun_x(Xo - h) - (2 * fun_x(Xo)) + 
fun_x(Xo + h));
                    cout << "f``(" << Xo << ")   =  " << f2x << endl;
                    break;
                case 5:
                    cout << "\n" << "Enter the value of Xo in f`(Xo) and f``(Xo):" 
<< endl;
                    cin >> Xo;
                    cout << "The following approximation is calculated using 
three-point midpoint formula." << endl;
                    h = (y - x) / (double)(datapt - 1);
                    //for (int i = 1; i < datapt - 2; i++){}
                    f1x = (1 / (2 * h))*(fun_x(Xo + h) - fun_x(Xo - h));
                    cout << "f`(" << Xo << ")   =  " << f1x << endl;
                    f2x = (1 / (h * h))*(fun_x(Xo - h) - (2 * fun_x(Xo)) + 
fun_x(Xo + h));
                    cout << "f``(" << Xo << ")   =  " << f2x << endl;
                    break;
                default:
                    temp = 0;
                    break;
                }
            }
            cout << endl << "To continue enter 1 else enter 0" << endl;
            cin >> mov;
        }

        return 0;
}
```

```
Output: console screen
```

Divyashree H B

Newtons differential differance

Enter N to give functional values else enter Y

N

Enter the intervals

0   1.1

Enter the number of data points

6

Enter 1 to print the difference table

2 To compute interpolating polynomial

3 For computing first derivative

4 For computing secound derivative

5 For computing both derivative

Anything else to break :

2

Enter the values :

-6.0

-5.89483

-5.65014

-5.17788

-4.28172

-3.99583

Differantial Table :

0.000000 -6.000000 0.525850 1.744000 1.834375 2.819792 -34.864062

0.200000 -5.894830 1.223450 2.844625 4.090208 -32.044271

0.400000 -5.650140 2.361300 5.298750 -21.545208

0.600000 -5.177880 4.480800 -7.628375

0.800000 -4.281720 1.429450

1.000000 -3.995830

Enter the value of X:

0.34

P(0.340000) = -5.729433

Enter 1 to print the difference table

2 To compute interpolating polynomial

3 For computing first derivative

4 For computing secound derivative

5 For computing both derivative

Anything else to break :

5

Enter the value of Xo in f\`(Xo) and f\`\`(Xo):

0.6

The following approximation is calculated using three-point midpoint formula.

f\`(0.600000) = 1.826334

f\`\`(0.600000) = 5.116125

Enter 1 to print the difference table

2 To compute interpolating polynomial

3 For computing first derivative

4 For computing secound derivative

5 For computing both derivative

Anything else to break :

6

To continue enter 1 else enter 0

1

Enter N to give functional values else enter Y

Y

Enter the intervals

-1 1

Enter the number of data points

11

Enter 1 to print the difference table

2 To compute interpolating polynomial

3 For computing first derivative

4 For computing secound derivative

5 For computing both derivative

Anything else to break :

3

Enter the value of Xo in f`(Xo) :

-0.4

The following approximation is calculated using three-point midpoint formula.

f`(-0.400000)   = -0.981297

Enter 1 to print the difference table

    2 To compute interpolating polynomial

    3 For computing first derivative

    4 For computing secound derivative

    5 For computing both derivative

    Anything else to break :

6

//screen shot

C:\Users\divya\OneDrive\documents\visual studio 2015\Projects\2a\Debug\2a.exe

```
Divyashree H B
Newtons differential differance

Enter N to give functional values else enter Y
N

Enter the intervals
0   1.1

Enter the number of data points
6

Enter 1 to print the difference table
      2 To compute interpolating polynomial
      3 For computing first derivative
      4 For computing secound derivative
      5 For computing both derivative
      Anything else to break :
2

Enter the values :
-6.0
-5.89483
-5.65014
-5.17788
-4.28172
-3.99583

Differantial Table :
0.000000  -6.000000  0.525850  1.744000  1.834375  2.819792  -34.864062
0.200000  -5.894830  1.223450  2.844625  4.090208  -32.044271
0.400000  -5.650140  2.361300  5.298750  -21.545208
0.600000  -5.177880  4.480800  -7.628375
0.800000  -4.281720  1.429450
1.000000  -3.995830

Enter the value of X:
0.34
P(0.340000)    =  -5.729433


Enter 1 to print the difference table
      2 To compute interpolating polynomial
      3 For computing first derivative
```

C:\Users\divya\OneDrive\documents\visual studio 2015\Projects\2a\Debug\2a.exe

```
Enter the value of X:
0.34
P(0.340000)   =  -5.729433


Enter 1 to print the difference table
      2 To compute interpolating polynomial
      3 For computing first derivative
      4 For computing secound derivative
      5 For computing both derivative
      Anything else to break :
5

Enter the value of Xo in f`(Xo) and f``(Xo):
0.6
The following approximation is calculated using three-point midpoint formula.
f`(0.600000)   =  1.826334
f``(0.600000)   =  5.116125

Enter 1 to print the difference table
      2 To compute interpolating polynomial
      3 For computing first derivative
      4 For computing secound derivative
      5 For computing both derivative
      Anything else to break :
6

To continue enter 1 else enter 0
1

Enter N to give functional values else enter Y
Y

Enter the intervals
-1  1

Enter the number of data points
11

Enter 1 to print the difference table
      2 To compute interpolating polynomial
      3 For computing first derivative
      4 For computing secound derivative
```

C:\Users\divya\OneDrive\documents\visual studio 2015\Projects\2a\Debug\2a.exe

```
      Anything else to break :
6

To continue enter 1 else enter 0
1

Enter N to give functional values else enter Y
Y

Enter the intervals
-1  1

Enter the number of data points
11

Enter 1 to print the difference table
      2 To compute interpolating polynomial
      3 For computing first derivative
      4 For computing secound derivative
      5 For computing both derivative
      Anything else to break :
3

Enter the value of Xo in f`(Xo) :
-0.4
The following approximation is calculated using three-point midpoint formula.
f`(-0.400000)   =  -0.981297

Enter 1 to print the difference table
      2 To compute interpolating polynomial
      3 For computing first derivative
      4 For computing secound derivative
      5 For computing both derivative
      Anything else to break :
6

To continue enter 1 else enter 0
```