NM

Solution of Lab Assignment -3 ID:C221050

```
/*


3.1 Find the first and second derivative at x = 1
   using Newton's Forward Difference formula.

x        1   2   3   4   5
y=f(x)    1   8   27  64  125


*/

#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    int n;
    cout << "Enter number of points (n): ";
    cin >> n;

    double x[100], y[100], diff[100][100];


    cout << "Enter x values:\n";
    for (int i = 0; i < n; ++i)
        cin >> x[i];
```

```
cout << "Enter y = f(x) values:\n";
for (int i = 0; i < n; ++i)
    cin >> diff[i][0];  // y

// forward difference table
for (int j = 1; j < n; ++j)
{
    for (int i = 0; i < n - j; ++i)
    {
        diff[i][j] = diff[i + 1][j - 1] - diff[i][j - 1];
    }
}

double X;
cout << "Enter value of X : ";
cin >> X;

double h = x[1] - x[0];
double u = (X - x[0]) / h;

// Factorial
double f2 = 2, f3 = 6, f4 = 24;

// First derivative
double f1 = (1 / h) * (
                diff[0][1] +
                ((2*u - 1) * diff[0][2]) / f2 +
                ((3*u*u - 6*u + 2) * diff[0][3]) / f3 +
                ((4*u*u*u - 18 * u * u + 22 * u - 6) * diff[0][4]) / f4
            );

// Second derivative
double f2nd = (1 / (h * h)) * (
                  diff[0][2] +
                    ((6*u - 6) * diff[0][3]) / f3 +
                    ((12*u*u - 36*u + 22) * diff[0][4]) / f4
                );

    // cout << fixed << setprecision(4);
    cout << "\nf'(X)  = " << f1 << endl;
    cout << "f''(X) = " << f2nd << endl;

    return 0;
}
/*
Input:
5
1 2 3 4 5
1 8 27 64 125
1
Output:
f'(X)  = 3
f''(X) = 6


input:
5
2 4 6 8 10
1.23 4.11 6.23 10.34 12.34
4
output:
f'(X)  = 0.735417
f''(X) = -0.0472917
Input:
6
1 2 3 4 5 6
1 8 27 64 125 216
1
Output:
f'(X)  = 3
```

```
f''(X) = 6
*/
```

```
/*
3.2 The following values of f (x) are given.
     x              1     2     3       4        5
   y = f(x)     1     8    27     64     125
Write a program to find the first derivative and the second derivative of the fu
nction tabulated above at the point x = 1.5.

*/

#include<bits/stdc++.h>

using namespace std;
int main()
{
    int n;
    cout << "Enter number of points (n): ";
    cin >> n;

    double x[100], y[100], diff[100][100];



    cout << "Enter x values:\n";
    for (int i = 0; i < n; ++i)
        cin >> x[i];



    cout << "Enter y = f(x) values:\n";
    for (int i = 0; i < n; ++i)
        cin >> diff[i][0];  // y

    // forward difference table
```

```
for (int j = 1; j < n; ++j)
{
    for (int i = 0; i < n - j; ++i)
    {
        diff[i][j] = diff[i + 1][j - 1] - diff[i][j - 1];
    }
}

double X;
cout << "Enter value of X : ";
cin >> X;

double h = x[1] - x[0];
double u = (X - x[0]) / h;

// Factorial
double f2 = 2, f3 = 6, f4 = 24;

// First derivative
double f1 = (1 / h) * (
                diff[0][1] +
                ((2*u - 1) * diff[0][2]) / f2 +
                ((3*u*u - 6*u + 2) * diff[0][3]) / f3 +
                ((4*u*u*u - 18 * u * u + 22 * u - 6) * diff[0][4]) / f4
            );

// Second derivative
double f2nd = (1 / (h * h)) * (
                  diff[0][2] +
                  ((6*u - 6) * diff[0][3]) / f3 +
                  ((12*u*u - 36*u + 22) * diff[0][4]) / f4
              );

// cout << fixed << setprecision(4);
cout << "\nf'(X)  = " << f1 << endl;
cout << "f''(X) = " << f2nd << endl;
```

```
    return 0;
}
/*

input:
5
1 2 3 4 5
1 8 27 64 125
1.5
Output:
f'(X)  = 6.75
f''(X) = 9
*/
```

```
/*
3.3 Write a program to calculate the approximate area under the curve y = ∫(1 to 5) log10 x dx                                   by using trapezoidal rule.
Theory:
I =h/2 [y0 + 2 (y1 + y2 + ..........  + yn-1) + yn]
*/



#include<bits/stdc++.h>

using namespace std;
double f(double x) {
    return log10(x);
}

int main() {
    double a, b;
    int n;

    cout << "Enter lower limit (a): ";
    cin >> a;
    cout << "Enter upper limit (b): ";
    cin >> b;
    cout << "Enter number of intervals (n): ";
    cin >> n;

    if (a <= 0 || b <= 0) {
        cout << "Error: log10(x) is undefined for x <= 0" << endl;
        return 1;
    }

    double h = (b - a) / n;
    double sum = f(a) + f(b);

    for (int i = 1; i < n; i++) {
        double x = a + i * h;
        sum += 2 * f(x);
    }

    double result = (h / 2) * sum;

    cout << "\nApproximate area under log10(x) from " << a << " to " << b << " is: " << result << endl;

    return 0;
}
/*
Input:
1
5
10
Output:
1.75307
*/
```

```
/*
3.4 Write a program to calculate the approximate area under the curve y = ∫(0 to pi/2) esinx dx by using Simpson’s 1/3 rule

*/



#include<bits/stdc++.h>

using namespace std;



double f(double x) {
    return exp(sin(x));  // e^sin(x)
}

int main() {
    double a, b;
    int n;

    cout << "Enter lower limit (a in radians): ";
    cin >> a;
    cout << "Enter upper limit (b in radians): ";
    cin >> b;
    cout << "Enter number of intervals (n - must be even): ";
    cin >> n;

    if (n % 2 != 0) {
        cout << "Error: Number of intervals must be even for Simpson’s 1/3 Rule." << endl;
        return 1;
    }

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

    double result = (h / 3) * sum;

    cout << "\nApproximate area under e^sin(x) from " << a << " to " << b << " is: " << result << endl;

    return 0;
}
/*
Input:
0
1.5708
10
Output:
3.10439
*/
```

```
/*
3.5 Write a program to calculate the approximate area under the curve:

    y = ∫(x / (1 + x^2)) dx from a to b

using Simpson’s 3/8 Rule.

Formula:
   I = (3h/8) * [(y0 + y_n) + 3(y1+y2+y4+y6+...+y_n-2+y_n-1) ) + 2(y3+y6+... + y_n-3) ]
```

```
where:
    h = (b - a) / n
    n must be a multiple of 3
*/

#include<bits/stdc++.h>
using namespace std;


double f(double x) {
    return x / (1 + x * x); // f(x) = x / (1 + x^2)
}

int main() {
    double a, b;
    int n;

    cout << "Enter lower limit (a): ";
    cin >> a;
    cout << "Enter upper limit (b): ";
    cin >> b;
    cout << "Enter number of intervals (n - must be multiple of 3): ";
    cin >> n;

    if (n % 3 != 0) {
        cout << "Error: Number of intervals must be a multiple of 3 for Simpson’s 3/8 Rule." << endl;
        return 1;
    }

    double h = (b - a) / n;
    double sum = f(a) + f(b);

    for (int i = 1; i < n; i++) {
        double x = a + i * h;
        if (i % 3 == 0)
```

```
            sum += 2 * f(x);
        else
            sum += 3 * f(x);
    }

    double result = (3 * h / 8) * sum;

    cout << "\nApproximate area under x / (1 + x^2) from " << a << " to " << b << " is: " << result << endl;

    return 0;
}

/*
Input:
0
1
30
Output:
0.346574
*/
```

```
/*
3.6 Write a program to find the determinant of a 3×3 matrix.
*/

#include <iostream>
using namespace std;

int main() {
    double matrix[3][3];

    cout << "Enter the elements of the 3×3 matrix row wise:\n";
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++) {
            cout << "Element [" << i+1 << "][" << j+1 << "]: ";
            cin >> matrix[i][j];
        }

    double a = matrix[0][0], b = matrix[0][1], c = matrix[0][2];
    double d = matrix[1][0], e = matrix[1][1], f = matrix[1][2];
    double g = matrix[2][0], h = matrix[2][1], i = matrix[2][2];

    double determinant = a*(e*i - f*h) - b*(d*i - f*g) + c*(d*h - e*g);

    cout << "\nDeterminant of the matrix is: " << determinant << endl;

    return 0;
}


/*
input:
1 2 3
4 5 6
7 8 9
output: 0
*/
```

```
/*
3.7 Solve 3×3 linear system using Matrix Inversion Method

Equations:
a11*x + a12*y + a13*z = b1
a21*x + a22*y + a23*z = b2
a31*x + a32*y + a33*z = b3

X = A^-1 * B
*/
```

```
#include <iostream>
using namespace std;

int main() {
    double A[3][3], B[3];


    cout << "Enter coefficients of matrix A (3×3):\n";
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++) {
            cout << "A[" << i+1 << "][" << j+1 << "]: ";
            cin >> A[i][j];
        }


    cout << "Enter values of matrix B (3×1):\n";
    for (int i = 0; i < 3; i++) {
        cout << "B[" << i+1 << "]: ";
        cin >> B[i];
    }

    //determinant
    double det =
        A[0][0]*(A[1][1]*A[2][2] - A[1][2]*A[2][1]) -
        A[0][1]*(A[1][0]*A[2][2] - A[1][2]*A[2][0]) +
        A[0][2]*(A[1][0]*A[2][1] - A[1][1]*A[2][0]);

    if (det == 0) {
        cout << "Matrix is singular. No unique solution.\n";
        return 1;
    }

    //inverse of A (adjoint / det)
    double inv[3][3];
    inv[0][0] =  (A[1][1]*A[2][2] - A[1][2]*A[2][1]) / det;
    inv[0][1] = -(A[0][1]*A[2][2] - A[0][2]*A[2][1]) / det;
    inv[0][2] =  (A[0][1]*A[1][2] - A[0][2]*A[1][1]) / det;

    inv[1][0] = -(A[1][0]*A[2][2] - A[1][2]*A[2][0]) / det;
    inv[1][1] =  (A[0][0]*A[2][2] - A[0][2]*A[2][0]) / det;
    inv[1][2] = -(A[0][0]*A[1][2] - A[0][2]*A[1][0]) / det;

    inv[2][0] =  (A[1][0]*A[2][1] - A[1][1]*A[2][0]) / det;
    inv[2][1] = -(A[0][0]*A[2][1] - A[0][1]*A[2][0]) / det;
    inv[2][2] =  (A[0][0]*A[1][1] - A[0][1]*A[1][0]) / det;

    //  A^-1 * B
    double x[3] = {0, 0, 0};
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
            x[i] += inv[i][j] * B[j];


    cout << "\nSolution:\n";
    cout << "x = " << x[0] << endl;
    cout << "y = " << x[1] << endl;
    cout << "z = " << x[2] << endl;

    return 0;
}

/*
Input:
A =
    3 1 2
    2 -3 -1
    1 2 1
B =
    3 -3 4
Output:
```

```
x = 1
y = 2
z = -1

input:
1 1 1
1 2 4
1 3 4
1 6 6
Output:
x = -0.666667
y = 0
z = 1.66667
*/
```

```
/*
3.8 Write a program to solve the following system of linear equations by using
Cramer’s Rule:
          27x +   6y – z     =  85
          6x + 15y + 2z   = 72
          x +     y + 54z = 110
*/

#include<bits/stdc++.h>

using namespace std;

int main() {
    double A[3][3], B[3];



    cout << "Enter coefficient matrix A (3×3) row by row:\n";
    for(int i = 0; i < 3; ++i)
      for(int j = 0; j < 3; ++j) {
        cout << "A[" << i+1 << "][" << j+1 << "]: ";
    cin >> A[i][j];
  }


cout << "Enter right-hand side vector B (3×1):\n";
for(int i = 0; i < 3; ++i) {
  cout << "B[" << i+1 << "]: ";
  cin >> B[i];
}

auto det = [&](double M[3][3]) {
  return
    M[0][0] * (M[1][1]*M[2][2] - M[1][2]*M[2][1]) -
    M[0][1] * (M[1][0]*M[2][2] - M[1][2]*M[2][0]) +
    M[0][2] * (M[1][0]*M[2][1] - M[1][1]*M[2][0]);
};

double D = det(A);

if (D == 0) {
  cout << "no solution.\n";
  return 1;
}


double x[3][3], y[3][3], z[3][3];

for(int i = 0; i < 3; ++i) {
  // Column 0 swapped for Dx
  x[i][0] = B[i];
  x[i][1] = A[i][1];
  x[i][2] = A[i][2];

  // Column 1 swapped for Dy
  y[i][0] = A[i][0];
  y[i][1] = B[i];
```

```
      y[i][2] = A[i][2];

      // Column 2 swapped for Dz
      z[i][0] = A[i][0];
      z[i][1] = A[i][1];
      z[i][2] = B[i];
    }




    double Dx = det(x);
    double Dy = det(y);
    double Dz = det(z);


    double xx = Dx / D;
    double yy = Dy / D;
    double zz = Dz / D;

    // cout << fixed << setprecision(4);
    cout << "\nSolution:\n";
    cout << "x = " << xx << "\n";
    cout << "y = " << yy << "\n";
    cout << "z = " << zz << "\n";
    return 0;
}

/*
input:
6 1 -3
1 3 -2
2 1 4
5 5 8
output
x = 1
```

```
y = 2
z = 1

input
27 6 -1
6 15 2
1 1 54
85 72 110
output
x = 2.42548
y = 3.57302
z = 1.92595

*/
```

```
/*
3.9 Write a program to solve the following system of linear equations by using
Gaussian Elimination method.
          2x +  y  +   z = 10
           x  + 4y + 9z = 16
          3x + 2y + 3z = 18
*/

#include<bits/stdc++.h>

using namespace std;

int main() {

    double M[3][4];


    cout << "Enter coefficients and RHS for each equation:\n";
    for (int i = 0; i < 3; ++i) {
```

```
    cout << "Equation " << i+1 << ":\n";
    for (int j = 0; j < 4; ++j) {
        if (j < 3)
            cout << "  A[" << i+1 << "][" << j+1 << "]: ";
        else
            cout << "  b[" << i+1 << "]: ";
        cin >> M[i][j];
    }
}


for (int p = 0; p < 2; ++p) {
    for (int r = p+1; r < 3; ++r) {
        double factor = M[r][p] / M[p][p];
        for (int c = p; c < 4; ++c) {
            M[r][c] -= factor * M[p][c];
        }
    }
}


double x[3];
for (int i = 2; i >= 0; --i) {
    double sum = M[i][3];
    for (int j = i+1; j < 3; ++j) {
        sum -= (M[i][j] * x[j]);
    }
    x[i] = sum / M[i][i];
}


// cout << fixed << setprecision(6);
cout<< "\nSolution:\n";
cout << "x = " << x[0] << "\n";
cout << "y = " << x[1] << "\n";
cout << "z = " << x[2] << "\n";
```

```
    return 0;
}
/*
Input:
2 1 1 10
3 2 3 18
1 4 9 16
Output:
x1 = 7
x2 = -9
x3 = 5
Input:
2 1 1 10
1 4 9 16
3 2 3 18
*/
```