#include<bits/stdc++.h>

using namespace std;

double a(double x)

{

return x;

}

double b(double x)

{

return 3;

}

double c(double x)

{

return (1 + x);

}

double d(double x)

{

return (1 + x\*x);

}

double y\_prev(double x, double h)

{

return (a(x)/(h\*h) - b(x)/(2\*h));

}

double y\_curr(double x, double h)

{

return (c(x) - (2\*a(x))/(h\*h));

}

double y\_next(double x, double h)

{

return (a(x)/(h\*h) + b(x)/(2\*h));

}

vector <vector <double>> cofactor(vector <vector <double>> &mat, int x, int y)

{

int n = mat.size();

vector <vector <double>> temp;

for(int k=0; k<n; k++)

{

vector <double> temptemp;

for(int l=0; l<n; l++)

{

if(k != x && l != y) temptemp.push\_back(mat[k][l]);

}

if(temptemp.size() > 0) temp.push\_back(temptemp);

}

return temp;

}

double determinant(vector <vector <double>> &mat)

{

int n = mat.size();

if(n == 1) return mat[0][0];

double det = 0;

int sign = 1;

for(int i=0; i<mat.size(); i++)

{

vector <vector <double>> cf = cofactor(mat, i, 0);

det += sign\*mat[i][0]\*determinant(cf);

sign \*= -1;

}

return det;

}

vector <vector <double>> adjugate(vector <vector <double>> &mat)

{

int n = mat.size();

vector <vector <double>> cf(n, vector <double> (n));

int sign = 1;

for(int i=0; i<n; i++)

{

for(int j=0; j<n; j++)

{

vector <vector <double>> cof = cofactor(mat, i, j);

cf[j][i] = sign\*determinant(cof);

sign \*= -1;

}

}

return cf;

}

vector <vector <double>> matrixInverse(vector <vector <double>> &mat)

{

vector <vector <double>> adj = adjugate(mat);

double det = determinant(mat);

for(int i=0; i<mat.size(); i++)

{

for(int j=0; j<mat[i].size(); j++) adj[i][j] /= det;

}

return adj;

}

vector <vector <double>> matrixMultiplication(vector <vector <double>> &matA, vector <vector <double>> &matB)

{

int n = matA.size();

vector <vector <double>> res;

for(int i=0; i<n; i++)

{

double x = 0.0;

for(int j=0; j<n; j++) x += matA[i][j]\*matB[j][0];

res.push\_back({x});

}

return res;

}

void display(vector <vector <double>> &mat)

{

for(int i=0; i<mat.size(); i++)

{

for(int j=0; j<mat[i].size(); j++) cout << mat[i][j] << "\t\t";

cout << endl;

}

}

int main()

{

double x\_start, x\_end;

cout << "Enter the starting value of x : ";

cin >> x\_start;

cout << "Enter the ending value of x : ";

cin >> x\_end;

double y\_start, y\_end;

cout << "Enter the starting value of y : ";

cin >> y\_start;

cout << "Enter the ending value of y : ";

cin >> y\_end;

double h;

cout << "Enter h : ";

cin >> h;

vector <double> x\_value;

for(auto i=x\_start; i<=x\_end; i+=h) x\_value.push\_back(i);

vector <vector <double>> coeff;

vector <vector <double>> constant;

for(int i=1; i<=x\_value.size()-2; i++)

{

vector <double> temp(x\_value.size()-2);

double p = y\_prev(x\_value[i], h);

double q = y\_curr(x\_value[i], h);

double r = y\_next(x\_value[i], h);

double s = d(x\_value[i]);

if(i == 1)

{

constant.push\_back({s - p\*y\_start});

temp[i-1] = q;

temp[i] = r;

}

else if(i == x\_value.size() - 2)

{

constant.push\_back({s - r\*y\_end});

temp[i - 2] = p;

temp[i - 1] = q;

}

else

{

constant.push\_back({s});

temp[i - 2] = p;

temp[i - 1] = q;

temp[i] = r;

}

coeff.push\_back(temp);

}

cout << "Coefficient matrix for solving the equations : " << endl;

display(coeff);

cout << "Constant term matrix for solving the equations : " << endl;

display(constant);

vector <vector <double>> invCoeff = matrixInverse(coeff);

vector <vector <double>> solution = matrixMultiplication(invCoeff, constant);

cout << "Final solutions : " << endl;

display(solution);

return 0;

}

// OUTPUT

// Enter the starting value of x : 0

// Enter the ending value of x : 4

// Enter the starting value of y : 1

// Enter the ending value of y : 0

// Enter h : 1

// Coefficient matrix for solving the equations :

// 0 2.5 0

// 0.5 -1 3.5

// 0 1.5 -2

// Constant term matrix for solving the equations :

// 2.5

// 5

// 10

// Final solutions :

// 41.75

// 1

// -4.25