**LAB REPORT**

**OF**

**NUMERICAL METHODS**

Subject Code: CSC 212

**NAGARJUNA COLLEGE OF IT**

(Affiliated to Tribhuvan University)

Sankhamul, Lalitpur

**Submitted By: Submitted To:**

NAME : Krishtina Maharjan Arjun Lamichhane

ROLL NO : 9

SEMESTER : 3rd (Third)

PROGRAM : B.Sc.CSIT

**TABLE OF CONTENT**

|  |  |  |
| --- | --- | --- |
| **S.N.** | **TITLE** | **SIGNATURE** |
| 1 | Program to calculate the root of 4x3-2x+6 using bisection method |  |
| 2 | Program to calculate the root of x2-5x+6 using secant method |  |
| 3 | Program to calculate the root of x3-3x-2 using Newton-Raphson method |  |
| 4 | Program to calculate the root of x2-x-1 using fixed point method |  |
| 5 | Program to evaluate a polynomial using Horner’s method |  |
| 6 | Program to read a set of data points from user and compute interpolation value at specified point using Lagrange interpolation |  |
| 7 | Program to read a set of data points from user and compute interpolation value at specified point using Newton interpolation |  |
| 8 | Program to read a set of data points from user and fit the line Y = A + BX through the points by the method of least squares |  |
| 9 | Program to integrate a given function using trapezoidal rule |  |
| 10 | Program to integrate a given function using Simpsons 1/3 rule |  |
| 11 | Program to integrate a given function using Simpsons 3/8 rule |  |
| 12 | Program to solve system of nonlinear equations using Gauss-Elimination method |  |
| 13 | Program to solve system of nonlinear equations using Gauss Jordan method |  |
| 14 | Program to factorize matrix using Dolittle method |  |
| 15 | Program to factorize matrix using Cholesky method |  |
| 16 | Program to solve system of non-linear equations using Jacobi Iteration |  |
| 17 | Program to solve system of non-linear equations using Gauss-Seidel method |  |
| 18 | Program to find eigenvalue and eigenvector using power method |  |
| 19 | Program to implement Euler’s method method to solve ordinary differential equations |  |
| 20 | Program to implement Heun’s method to solve ordinary differential equations |  |
| 21 | Program to solve boundary value problem using shooting method |  |
| 22 | Program to solve Laplacian Equation |  |
| 23 | Program to solve Poisson's Equation |  |

**Source Code :**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#define f(x) (4\*pow(x,3)-2\*x+6)

int main(){

float a=0,b=0,error=0,m,mold;

int i=0;

printf("Input Interval: ");

scanf("%f %f",&a,&b);

if((f(a)\*f(b))>0){

printf("Incorrect initial guesses"); //to test whether search interval is okay or not exit(1);

}

else if(f(a)==0 || f(b)==0){

printf("Root is one of interval bounds. Root is %f\n",f(a)==0?a:b);

exit(0);

}

printf("Ite\ta\t\tb\t\tm\t\tf(m)\t\terror\n");

do{

mold=m;

m=(a+b)/2;

printf("%2d\t%4.6f\t%4.6f\t%4.6f\t%4.6f\t",i++,a,b,m,f(m));

if(f(m)==0){

printf("Root is %4.6f\n",m);

}else if ((f(a)\*f(m))<0){

b=m;

}else a=m;

error=fabs(m-mold);

if(i==1){

printf("----\n");

}else printf("%4.6f\n",error);

}while(error>0.001);

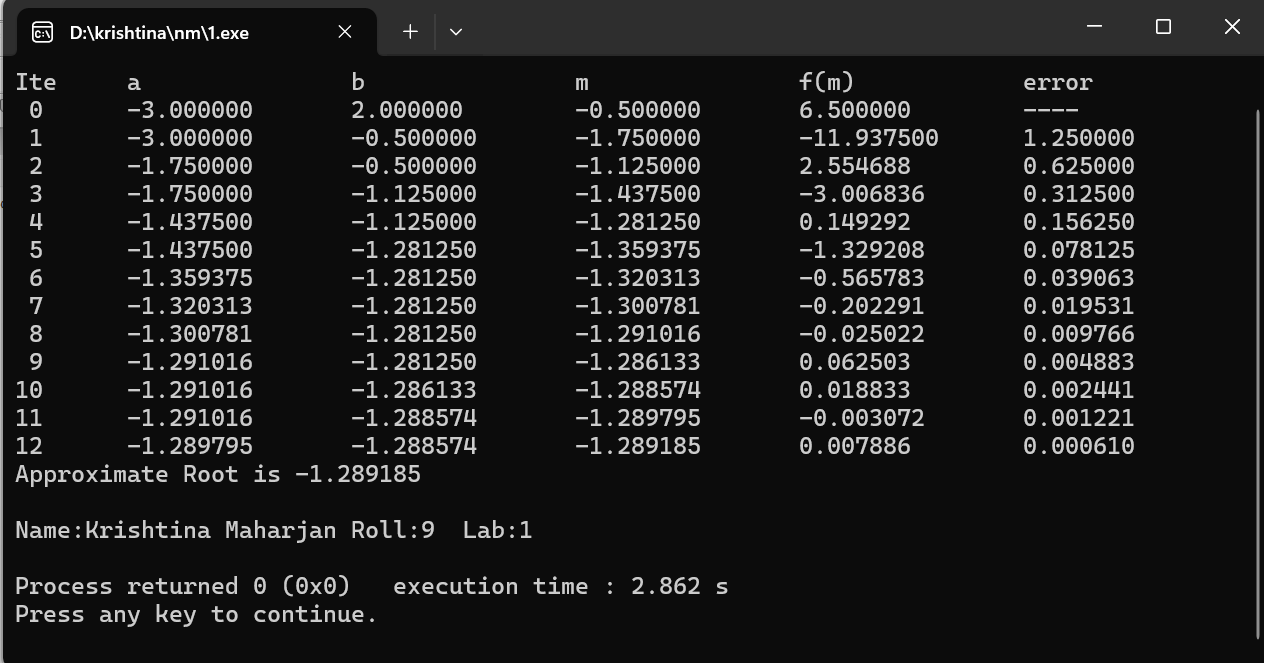
printf("Approximate Root is %4.6f\n",m);

printf("\nName:Krishtina Maharjan\tRoll:9\tLab:1\n");

return 0;

}

**Output :**



**Source Code :**

#include<stdio.h>

#include<math.h>

#define f(x) (pow(x,2)-5\*x+6)

int main(){

float x0,x1,x2,error;

int i=0;

printf("Input Two Approximations: ");

scanf("%f %f",&x0,&x1);

printf("Ite\tX0\t\tX1\t\tf(X0)\t\tf(X1)\t\tError\n");

do{

x2= x1-(f(x1)\*(x1-x0))/(f(x1)-f(x0));

printf("%2d\t%4.6f\t%4.6f\t%4.6f\t%4.6f\t%4.6f\n",i++,x0,x1,f(x0),f(x1),error);

error=fabs((x2)-(x1));

x0=x1;

x1=x2;

}while(error>0.00005);

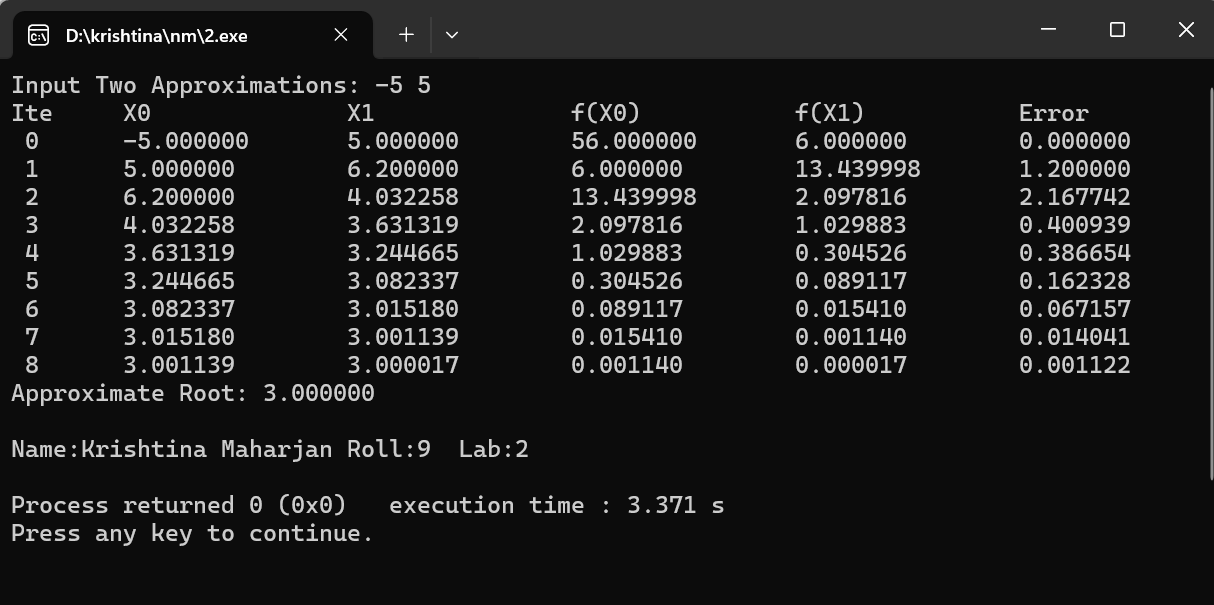
printf("Approximate Root: %4.6f\n", x2);

printf("\nName:Krishtina Maharjan\tRoll:9\tLab:2\n");

return 0;

}

**Output :**



**Source Code :**

#include <stdio.h>

#include<math.h>

#define EPS 0.000001 //Error Bound

#define MAXIT 20 //Maximum iterations

#define F(x) (x)\*(x)-(3\*x)+2 //function

#define FD(x) 2\*(x)-3 //1st derivative of function

int main(){

int count;

float x0, xn, fx, fdx, error, absError;

printf("\n");

printf("Input initial value of x: \n");

scanf("%f", &x0);

printf("\n");

printf("---SOLUTION BY NEWTON-RAPHSON METHOD---\n\n");

printf("x0 \t\t xn\t\t Error\t\t \n");

count = 1;

begin:

fx = F(x0);

fdx= FD(x0);

xn = x0 - fx/fdx; //new value of x

error=(xn-x0)/xn; //Error Calculation

absError=fabs(error);

printf("%f \t %f \t %f\n", x0, xn, absError);

if (absError<EPS){

printf("Root = %f \n", xn);

printf("Function value %f \n", F(xn));

printf ("Number of iterations = %d \n", count);

printf("\nName:Krishtina Maharjan\tRoll:9\tLab:3");

printf("\n");

}

else{

x0=xn; //set new value of x

count = count+1;

if(count<MAXIT){

goto begin;

}else{

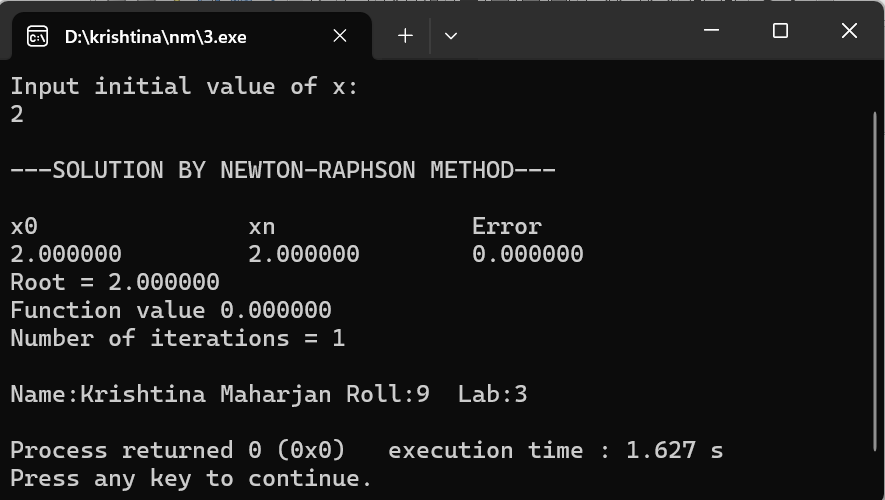
printf("Solution doesnot exist in %d iterations",MAXIT);

}

}

}

**Output :**



**Source Code :**

#include<stdio.h>

#include<conio.h>

#include<math.h>

#define f(x) (x\*x-x-1)          //function to be solved

#define g(x) (sqrt(x+1))       //Write f(x) as x = g(x) and

int main(){

     int step=1, N;

     float x0, x1, e;

     printf("Enter initial guess: ");

     scanf("%f", &x0);

     printf("Enter tolerable error: ");

     scanf("%f", &e);

     printf("Enter maximum iteration: ");

     scanf("%d", &N);

     printf("\nStep\tx0\t\tf(x0)\t\tx1\t\tf(x1)\n");    //Implementing Fixed Point Iteration

     do{

          x1 = g(x0);

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

          step = step + 1;

          if(step>N) {

               printf("Not Convergent.");

               exit(0);

          }

          x0 = x1;

        }while( fabs(f(x1)) > e);

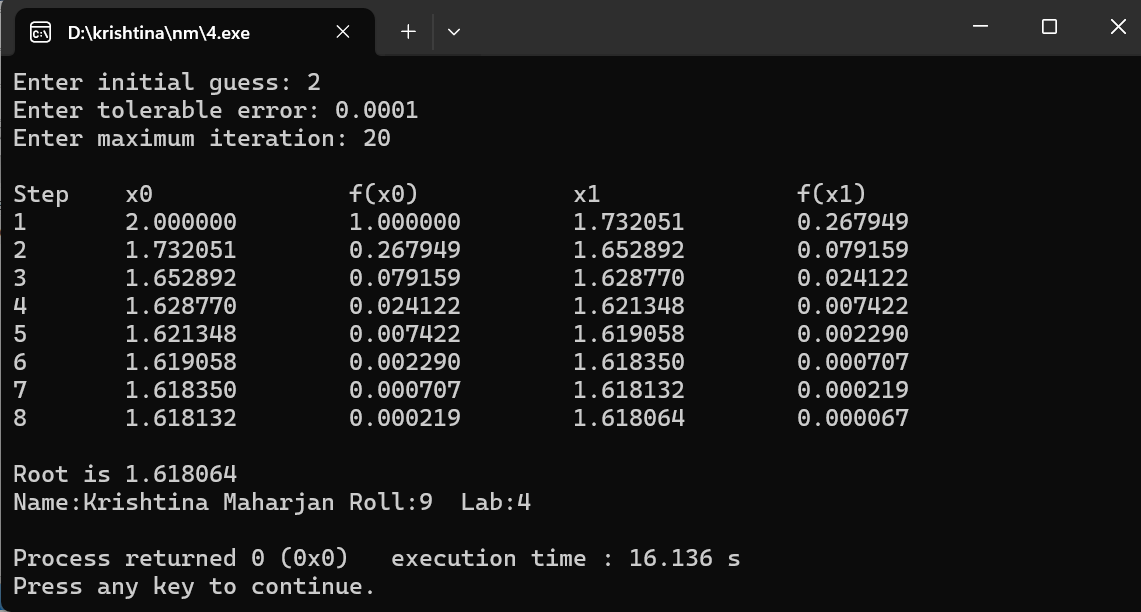
     printf("\nRoot is %f", x1);

printf("\nName:Krishtina Maharjan\tRoll:9\tLab:4\n");

     return(0);

}

**Output :**



**Source Code :**

#include <stdio.h>

double horner(int coeffs[], int n, int x) {

double result = coeffs[0];

for (int i = 1; i < n; i++)

result = result \* x + coeffs[i];

return result;

}

int main() {

int n, x;

printf("Enter the degree of the polynomial: ");

scanf("%d", &n);

int coeffs[n + 1];

printf("Enter the coefficients from highest to lowest degree:\n");

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

scanf("%d", &coeffs[i]);

printf("Enter the value of x: ");

scanf("%d", &x);

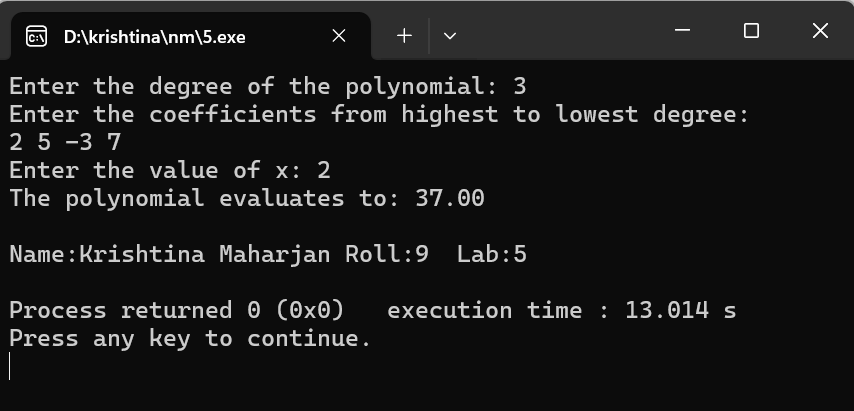
printf("The polynomial evaluates to: %.2f\n", horner(coeffs, n + 1, x));

printf("\nName:Krishtina Maharjan\tRoll:9\tLab:5\n");

return 0;

}

**Output :**



**Source Code :**

#include<stdio.h>

int main(){

int n;

printf("Enter no. of terms\n");

scanf("%d",&n);

float X[n],Y[n],x,sum=0,term;

int i,j;

printf("Enter Values of X \n");

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

scanf("%f",&X[i]);

printf("Enter Values of Y\n");

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

scanf("%f",&Y[i]);

printf("Enter value of x for which you want y\n");

scanf("%f",&x);

// Applying the Formula

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

term=1;

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

if(i!=j)

term = term \* ((x - X[j])/(X[i]-X[j]));

}

sum=sum + term \* Y[i];

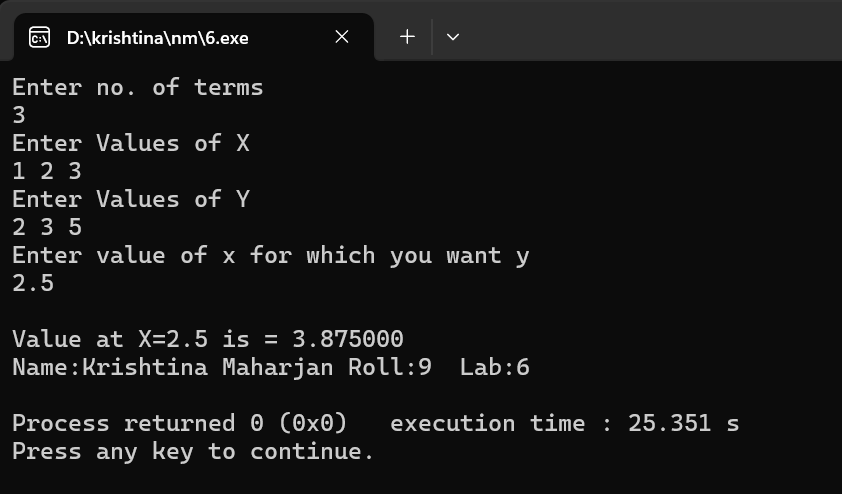
}

printf("\nValue at X=%g is = %f", x,sum);

printf("\nName:Krishtina Maharjan\tRoll:9\tLab:6\n");

}

**Output :**



**Source Code :**

#include<stdio.h>

#include<math.h>

int fact(int);

void main() {

float arr[10][11], x, h, p, y, px = 1;

int i, j, n, ch = 30;

printf("Enter the number of data: ");

scanf("%d", &n);

printf("\nEnter the data:\n");

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]);

}

// Constructing Difference Table

for(j = 2; j <= n; j++) {

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

arr[i][j] = arr[i + 1][j - 1] - arr[i][j - 1];

}

}

printf("\nDifference Table:\n");

printf("X\tY");

for(i = 1; i < n; i++) {

printf("\t%c^%dY", ch, i);

}

printf("\n");

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

printf("%.2f\t", arr[i][0]);

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

printf("%.4f\t", arr[i][j]);

}

printf("\n");

}

printf("\nEnter the value of x for function f(x): ");

scanf("%f", &x);

h = arr[1][0] - arr[0][0];

p = (x - arr[0][0]) / h;

y = arr[0][1];

for(i = 1; i < n; i++) {

px = px \* (p - (i - 1));

y = y + (arr[0][i + 1] \* px) / fact(i);}

printf("\nThe value of function at x = %f is %f", x, y);

printf("\nName: Krishtina Maharjan Roll: 09 Lab: 7\n");

}

int fact(int n) {

int i, f = 1;

for(i = 1; i <= n; i++) {

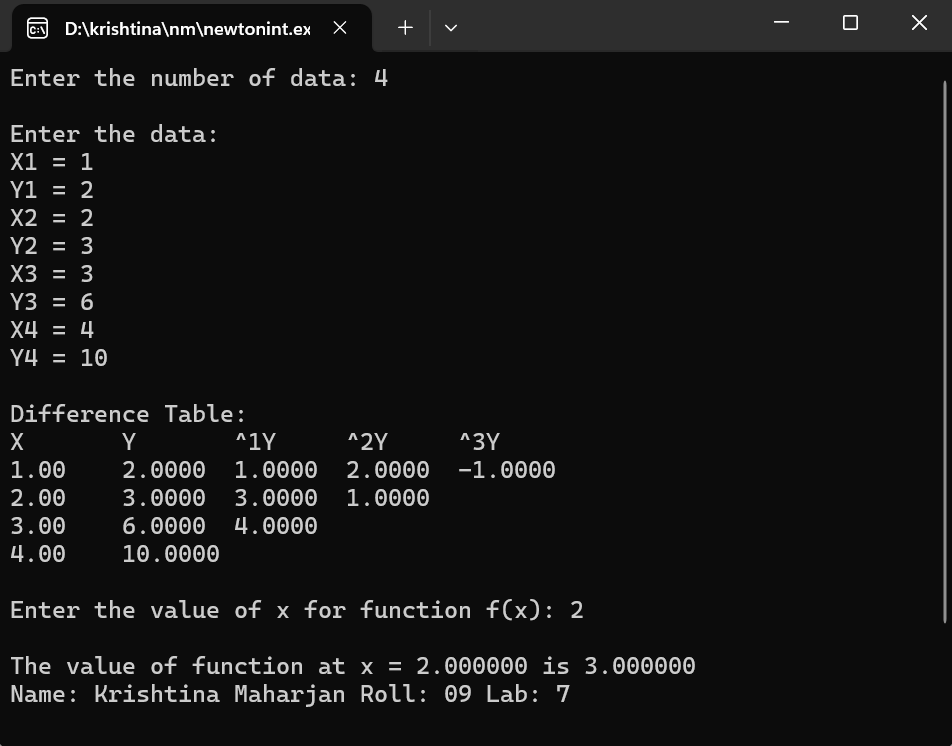
f = f \* i;

}

return f;

}

**Output :**



**Source Code :**

#include <stdio.h>

// Function to perform linear regression

void linearRegression(float x[], float y[], int n, float \*slope, float \*intercept) {

float sumX = 0.0, sumY = 0.0, sumXY = 0.0, sumX2 = 0.0;

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

sumX += x[i];

sumY += y[i];

sumXY += x[i] \* y[i];

sumX2 += x[i] \* x[i];

}

float numerator = n \* sumXY - sumX \* sumY;

float denominator = n \* sumX2 - sumX \* sumX;

\*slope = numerator / denominator;

\*intercept = (sumY - (\*slope) \* sumX) / n;

}

int main() {

int n;

printf("Enter the number of data points: ");

scanf("%d", &n);

float x[n], y[n];

printf("Enter data points in the format 'x y':\n");

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

scanf("%f %f", &x[i], &y[i]);

}

float slope, intercept;

linearRegression(x, y, n, &slope, &intercept);

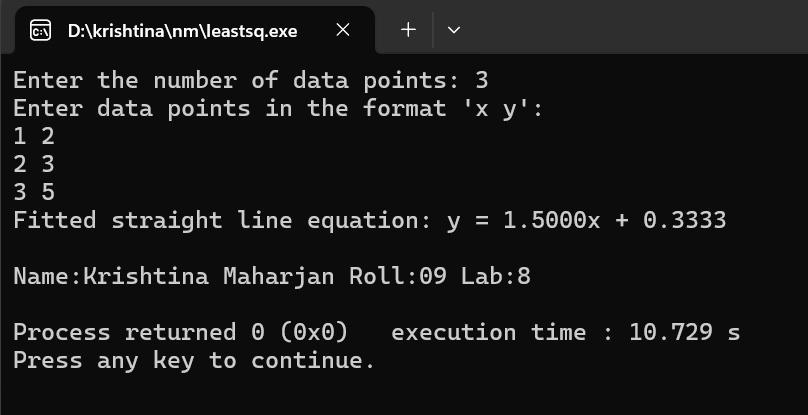
printf("Fitted straight line equation: y = %.4fx + %.4f\n", slope, intercept);

printf("\nName:Krishtina Maharjan Roll:09 Lab:8\n");

return 0;

}

**Output :**



**Source Code :**

//C Program to Implement Trapezoidal Rule

#include<stdio.h>

#define f(x) pow(x,3)+3\*x

float findValueAt(float x){

return f(x);

}

int main(){

int n;

float i,a,b,sum=0,h;

//Input

printf("Enter Value of a and b\n");

scanf("%f%f",&a,&b);

printf("Enter no. of Intervals\n");

scanf("%d",&n);

//....................Computations .....................

h=(b-a)/n;

sum = findValueAt(a) +findValueAt(b);

for(i=a+h;i<b;i=i+h)

sum = sum + 2\*findValueAt(i);

sum = (h \* sum)/2;

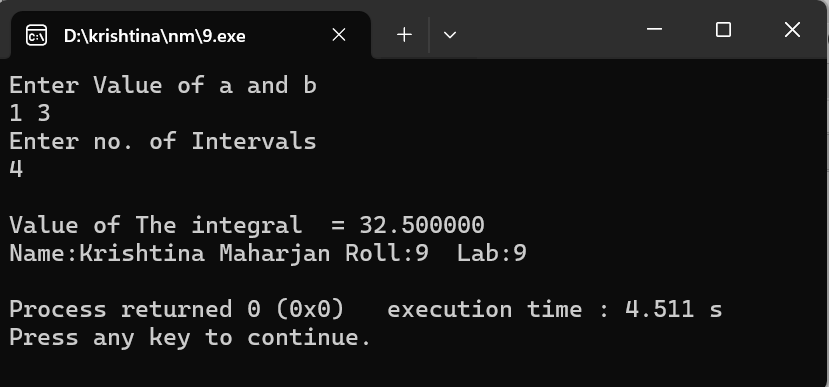
//Print the Output

printf("\nValue of The integral = %f",sum);

printf("\nName:Krishtina Maharjan\tRoll:9\tLab:9\n");

}

**Output :**



**Source Code :**

//C Program to Implement Simpson's 1/3 Rule

#include<stdio.h>

#define f(x) pow(x,3)+3\*x

float findValueAt(float x){

return f(x);

}

int main(){

int n;

float i,a,b,sum=0,h;

//The initial Position (0) is treated as Even position

int position\_of\_term=1;

printf("Enter Value of a and b\n");

scanf("%f%f",&a,&b);

printf("Enter no. of Intervals\n");

scanf("%d",&n);

//....................Computations .....................

h=(b-a)/n;

sum = findValueAt(a) +findValueAt(b);

for(i=a+h;i<b;i=i+h)

{

if(position\_of\_term %2 ==0)

sum = sum + 2\*findValueAt(i);

else

sum = sum + 4\*findValueAt(i);

position\_of\_term++;

}

sum = (h \* sum)/3;

//Print the Output

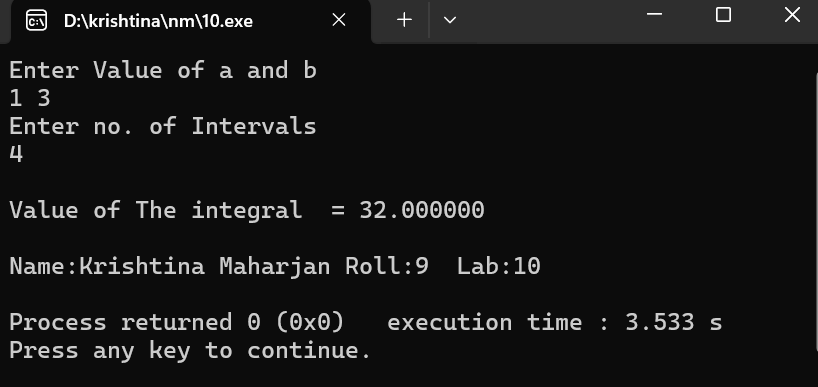
printf("\nValue of The integral = %f",sum);

printf("\n");

printf("\nName:Krishtina Maharjan\tRoll:9\tLab:10\n");

}

**Output :**



**Source Code :**

#include <stdio.h>

#include <math.h>

#define f(x) (pow(x,3)+3\*x)

int main() {

float lower, upper, integration = 0.0, stepSize, k;

int i, subInterval;

printf("Enter lower limit of integration: ");

scanf("%f", &lower);

printf("Enter upper limit of integration: ");

scanf("%f", &upper);

printf("Enter number of sub intervals: ");

scanf("%d", &subInterval);

stepSize = (upper - lower) / subInterval;

integration = f(lower) + f(upper);

for (i = 1; i <= subInterval - 1; i++) {

k = lower + i \* stepSize;

if (i % 3 == 0) {

integration = integration + 2 \* f(k);

} else {

integration = integration + 3 \* f(k);

}

}

integration = integration \* stepSize \* 3 / 8;

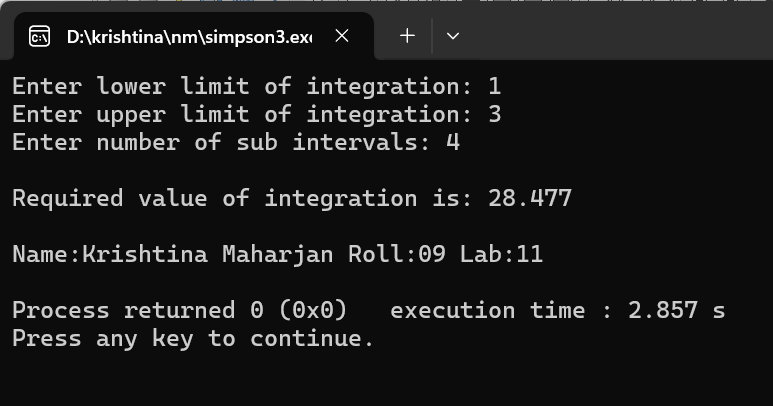
printf("\nRequired value of integration is: %.3f\n", integration);

printf("\nName:Krishtina Maharjan Roll:09 Lab:11\n");

return 0;

}

**Output :**



**Source Code :**

#include<stdio.h>

int n;

void convertToUpperTriangular(float a[][n+1],int n)

{

int i,j,x,y,k;

float ratio;

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

{

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

{

if(j>i)

{

ratio=a[j][i]/a[i][i];

for(k=0;k<n+1;k++)

a[j][k]=a[j][k] -( ratio \* a[i][k]);

printf("Intermediate forms:\n");

for( x=0;x<n;x++)

{

for(y=0;y<n+1;y++)

printf("%.2f ",a[x][y]);

printf("\n");

}

printf("\n");

}

}

}

}

void ApplyBackSubstitution(float a[][n+1],float value[],int n)

{

int i,j;

float sum;

value[n-1]=a[n-1][n]/a[n-1][n-1];

for(i=n-2;i>=0;i--)

{

sum=0;

for(j=i+1;j<n;j++)

sum=sum+a[i][j]\*value[j];

value[i] = (a[i][n]-sum)/a[i][i];

}

}

void print(float value[],int n)

{

int i;

printf("Values of unknowns are:\n");

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

printf("Value[%d]=%f\n",i,value[i]);

}

int main()

{

int i,j,k,x,y;

float sum,ratio;

printf("Enter no of Unknowns\n");

scanf("%d",&n);

float a[n][n+1],value[n];

printf("Enter the Augmented Matrix\n");

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

{

for(int j=0;j<n+1;j++)

scanf("%f",&a[i][j]);

}

convertToUpperTriangular(a,n);

ApplyBackSubstitution(a,value,n);

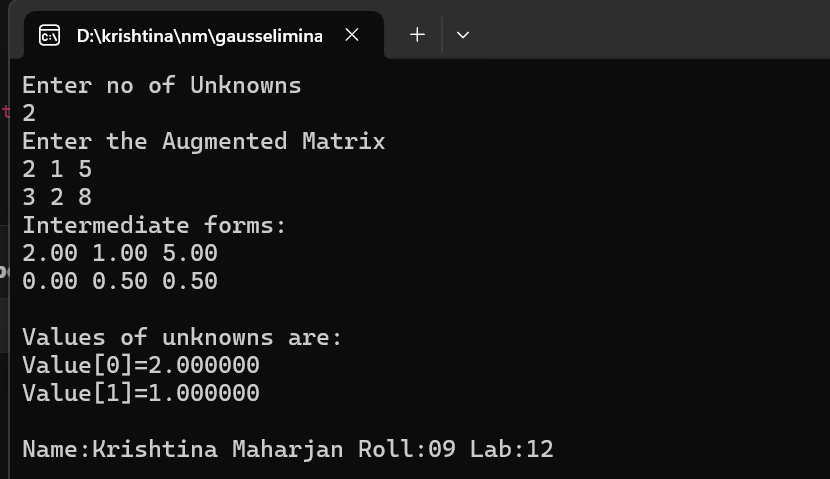
print(value,n);

printf("\nName:Krishtina Maharjan Roll:09 Lab:12\n");

return 0;

}

**Output :**



**Source Code :**

//C Program to Implement Gauss Jordan Method

#include <stdio.h>

int n;

void convertToDiagonal(float a[][n+1], int n) {

int i, j, x, y, k; float ratio;

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

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

if (j != i) {

ratio = a[j][i] / a[i][i];

for (k = 0; k < n + 1; k++) a[j][k] = a[j][k] - (ratio \* a[i][k]);

printf("Intermediate forms:\n");

for (x = 0; x < n; x++) {

for (y = 0; y < n + 1; y++) printf("%.3f ", a[x][y]);

printf("\n");

} printf("\n");

}

}

}

}

void printUnknowns(float a[][n+1], int n) {

int i; printf("Values of unknowns are:\n");

for (i = 0; i < n; i++) printf("Value of Variable %d = %f\n", i, a[i][n] / a[i][i]);

}

int main() {

int i, j, k, x, y; float ratio;

printf("Enter no of Unknowns\n"); scanf("%d", &n);

float a[n][n+1];

printf("Enter the Augmented Matrix\n");

for (i = 0; i < n; i++) for (j = 0; j < n + 1; j++) scanf("%f", &a[i][j]);

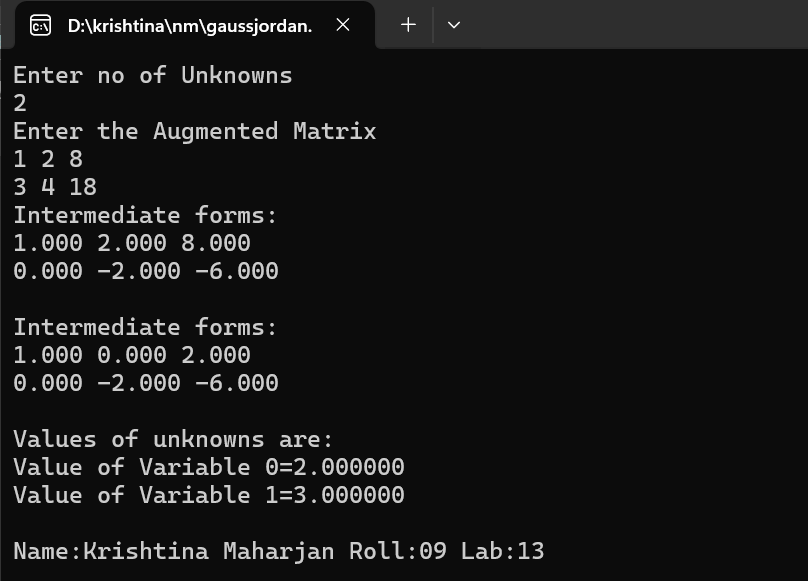
convertToDiagonal(a, n); printUnknowns(a, n);

printf("\nName: Krishtina Maharjan Roll: 09 Lab: 13\n");

return 0;

}

**Output :**



**Source Code :**

#include<stdio.h>

void main(){

float A[20][20]= {0},L[20][20]= {0}, U[20][20];

float B[20]= {0}, X[20]= {0},Y[20]= {0};

int i,j,k,n;

printf("Enter the order of square matrix: ");

scanf("%d",&n);

printf("\nEnter matrix element:\n");

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

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

printf("Enter A[%d][%d] element: ", i,j);

scanf("%f",&A[i][j]);

}

}

printf("\nEnter the constant terms: \n");

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

printf("B[%d]",i);

scanf("%f",&B[i]);

}

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

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

if(i<=j){

U[i][j]=A[i][j];

for(k=0; k<i-1; k++)

U[i][j]-=L[i][k]\*U[k][j];

if(i==j)

L[i][j]=1;

else

L[i][j]=0;

}

else{

L[i][j]=A[i][j];

for(k=0; k<=j-1; k++)

L[i][j]-=L[i][k]\*U[k][j];

L[i][j]/=U[j][j];

U[i][j]=0;

}

}

}

printf("[L]: \n");

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

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

printf("%9.3f",L[i][j]);

printf("\n");

}

printf("\n[U]: \n");

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

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

printf("%9.3f",U[i][j]);

printf("\n");

}

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

Y[i]=B[i];

for(j=0; j<i; j++){

Y[i]-=L[i][j]\*Y[j];

}

}

printf("\n[Y]: \n");

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

printf("%9.3f",Y[i]);

}

for(i=n-1; i>=0; i--){

X[i]= Y[i];

for(j=i+1; j<n; j++){

X[i]-=U[i][j]\*X[j];

}

X[i]/=U[i][i];

}

printf("\n[X]: \n");

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

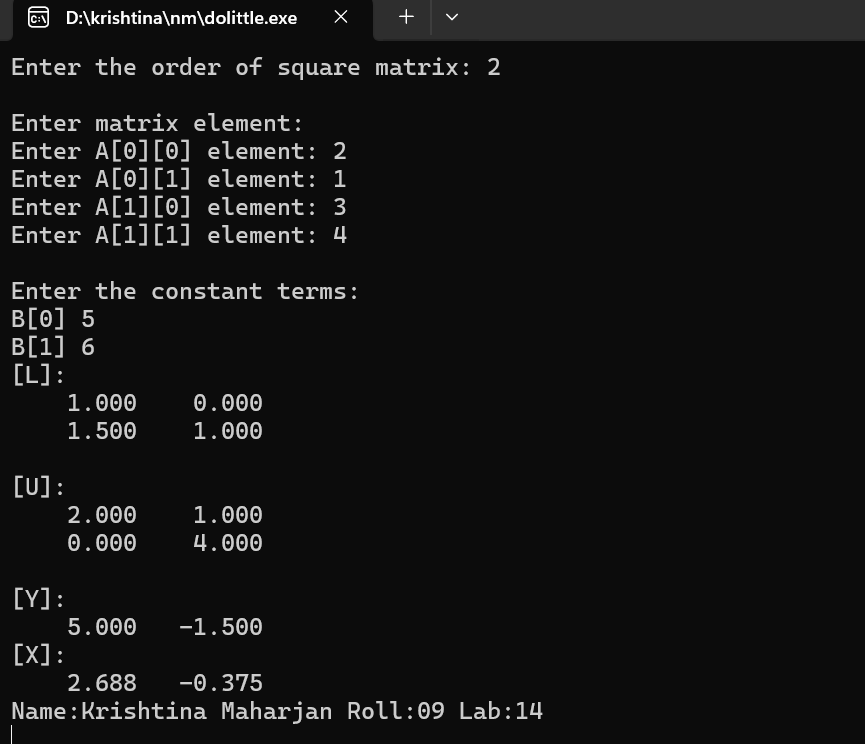
printf("%9.3f",X[i]);

}

printf("\nName:Krishtina Maharjan Roll:09 Lab:14\n");

}

**Output :**



**Source Code :**

#include <stdio.h>

#include <math.h>

void choleskyDecomposition(double A[][3], double L[][3], int n) {

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

for (int j = 0; j < (i+1); j++) {

double sum = 0;

if (i == j) {

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

sum += L[i][k] \* L[i][k];

L[i][j] = sqrt(A[i][i] - sum);

} else {

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

sum += L[i][k] \* L[j][k];

L[i][j] = (1.0 / L[j][j]) \* (A[i][j] - sum);

}

}

}

}

void printMatrix(double matrix[][3], int rows, int cols) {

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

printf("%.4f\t", matrix[i][j]);

}

printf("\n");

}

}

int main() {

int n;

printf("Enter the size of the matrix: ");

scanf("%d", &n);

double A[3][3];

printf("Enter the elements of the matrix:\n");

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

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

scanf("%lf", &A[i][j]);

}

}

double L[3][3] = {0};

choleskyDecomposition(A, L, n);

printf("\nOriginal Matrix A:\n");

printMatrix(A, n, n);

printf("\nCholesky Decomposition L:\n");

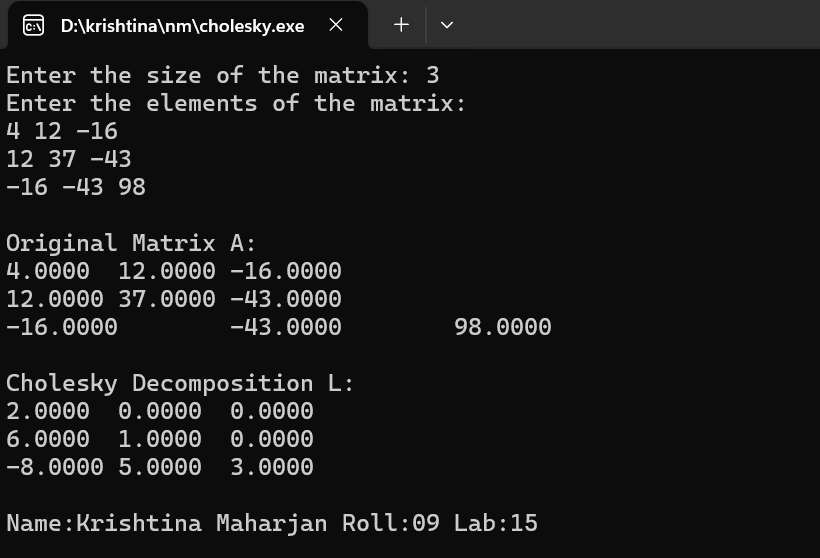
printMatrix(L, n, n);

printf("\nName:Krishtina Maharjan Roll:09 Lab:15\n");

return 0;

}

**Output :**



**Source Code :**

#include<stdio.h>

#include<math.h>

#include<stdbool.h>

#define EPSILON 0.001

int n;

int flag;

//Helper function

float findSum(int i,float a[][n+1]){

float sum=0;

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

if(i!=j)

sum+=a[i][j];

}

return sum;

}

//checks if Gauss Jacobi Method is applicable and return true if yes otherwise return false

bool isMethodApplicable(float a[][n+1]{

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

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

if(fabs(a[i][i])>findSum(i,a))

break;

else

return false;

}

}

return true;

}

//prints the Value of Unknowns

void print(int iteration,float values[n]){

printf("Iteration %d ",iteration);

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

printf("value[%d]=%f ",i+1,values[i]);

printf("\n");

}

void findValues(float a[][n+1],int maxIterations,float values\_old[n]){

int i,j,k,iteration;

float sum=0;

float values\_new[n];

for(iteration=1;iteration<=maxIterations;iteration++) {

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

sum=0;

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

if(i!=j)

sum+=a[i][j]\*values\_old[j];

}

values\_new[i]=(a[i][n] - sum)/a[i][i];

}

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

if(fabs(values\_old[k]-values\_new[k])<EPSILON) {

continue;

}

else{

flag=1;

break;

}

}

if(flag==0){

print(iteration,values\_new); // print final values of unknowns and return

return ;

}

flag=0; //resetting the flag

print(iteration,values\_new); //To print intermediate values of unknowns

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

values\_old[k]=values\_new[k];

}

print(iteration,values\_new) ;

} //end of findValues()

int main()

{

int i,j,k,x,y,maxIterations;

float ratio;

printf("Enter no of Unknowns\n");

scanf("%d",&n);

printf("Enter no. of iterations\n");

scanf("%d",&maxIterations);

float a[n][n+1];

float values[n];

printf("Enter the Augmented Matrix\n");

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

for(int j=0;j<n+1;j++)

scanf("%f",&a[i][j]);

}

if(!isMethodApplicable(a)){

printf("Gauss Jacobi Method can't be applied\n");

return 0;

}

printf("Gauss Jacobi Method is applicable\n");

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

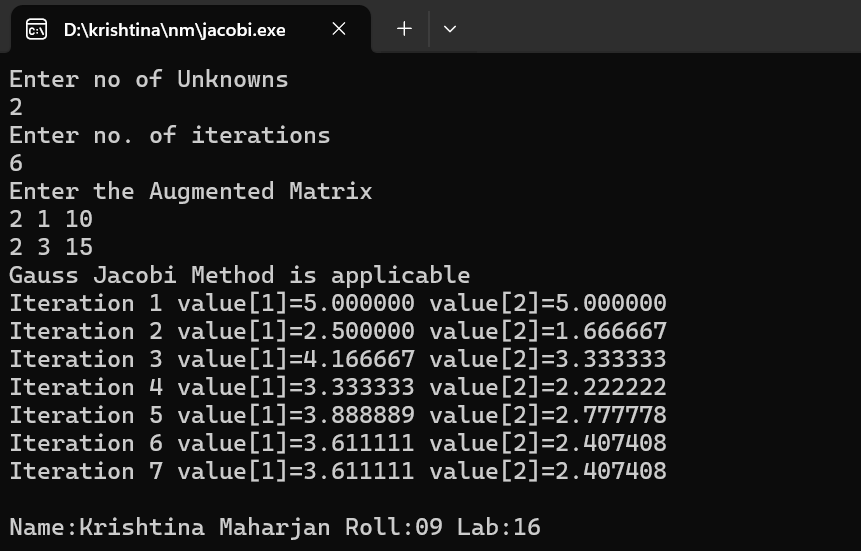
values[i]=0;

findValues(a,maxIterations,values);

printf("\nName:Krishtina Maharjan Roll:09 Lab:16\n");

return 0;

**Output :**



**Source Code :**

#include<stdio.h>

#include<math.h>

#include<stdbool.h>

#define EPSILON 0.001 // till 3 correct decimal places

int n, flag;

// Helper function

float findSum(int i, float a[][n+1]) {

float sum = 0;

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

if (i != j)

sum += a[i][j];

}

return sum;

}

// Checks if Gauss Seidel Method is applicable

bool isMethodApplicable(float a[][n+1]) {

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

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

if (fabs(a[i][i]) > findSum(i, a))

continue;

else

return false;

}

}

return true;

}

void print(int iteration, float values[n]) {

printf("Iteration %d ", iteration);

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

printf("value[%d]=%f ", i + 1, values[i]);

printf("\n");

}

void findValues(float a[][n+1], int maxIterations, float values\_old[n]) {

int i, j, k, iteration;

float sum = 0, values\_new[n];

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

values\_new[i] = 0;

for (iteration = 1; iteration <= maxIterations; iteration++) {

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

sum = 0;

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

if (i != j)

sum += a[i][j] \* values\_new[j];

}

values\_new[i] = (a[i][n] - sum) / a[i][i];

}

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

if (fabs(values\_old[k] - values\_new[k]) < EPSILON)

continue;

else {

flag = 1;

break;

}

}

if (flag == 0) {

print(iteration, values\_new);

return;

}

flag = 0;

print(iteration, values\_new);

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

values\_old[k] = values\_new[k];

}

print(iteration, values\_new);

}

int main() {

int i, j, maxIterations;

printf("Enter no of Unknowns\n");

scanf("%d", &n);

printf("Enter no. of iterations\n");

scanf("%d", &maxIterations);

float a[n][n+1], values[n];

printf("Enter the Augmented Matrix\n");

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

for (j = 0; j < n+1; j++)

scanf("%f", &a[i][j]);

}

if (!isMethodApplicable(a)) {

printf("Gauss Seidel Method can't be applied");

return 0;

}

printf("\n\nGauss Seidel Method is applicable\n");

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

values[i] = 0;

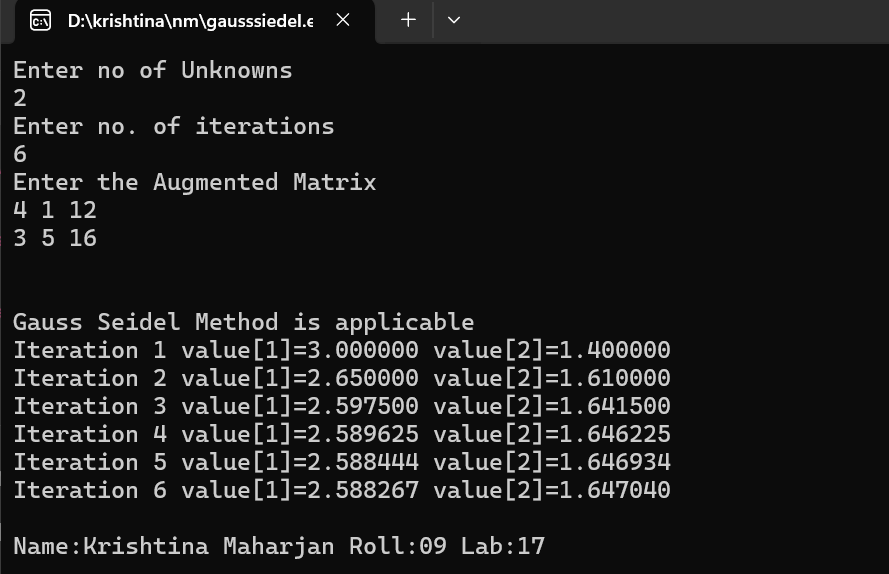
findValues(a, maxIterations, values);

printf("\nName: Krishtina Maharjan Roll: 09 Lab: 17\n");

return 0;

}

**Output :**



**Source Code :**

#include<stdio.h>

#include<conio.h>

#include<math.h>

void main() {

int i, j, n;

float A[40][40], x[40], z[40], e[40], zmax, emax;

printf("Enter the order of matrix:");

scanf("%d", &n);

printf("\nEnter matrix elements row-wise\n");

for (i = 1; i <= n; i++) {

for (j = 1; j <= n; j++) {

printf("A[%d][%d]=", i, j);

scanf("%f", &A[i][j]);

}

}

printf("Enter the column vector\n");

for (i = 1; i <= n; i++) {

printf("X[%d]=", i);

scanf("%f", &x[i]);

}

do {

for (i = 1; i <= n; i++) {

z[i] = 0;

for (j = 1; j <= n; j++) {

z[i] = z[i] + A[i][j] \* x[j];

}

}

zmax = fabs(z[1]);

for (i = 2; i <= n; i++) {

if ((fabs(z[i])) > zmax)

zmax = fabs(z[i]);

}

for (i = 1; i <= n; i++) {

z[i] = z[i] / zmax;

}

for (i = 1; i <= n; i++) {

e[i] = fabs((fabs(z[i])) - (fabs(x[i])));

}

emax = e[1];

for (i = 2; i <= n; i++) {

if (e[i] > emax)

emax = e[i];

}

for (i = 1; i <= n; i++) {

x[i] = z[i];

}

} while (emax > 0.001);

printf("\n The required eigen value is %f", zmax);

printf("\n\nThe required eigen vector is :\n");

for (i = 1; i <= n; i++) {

printf("%f\t", z[i]);

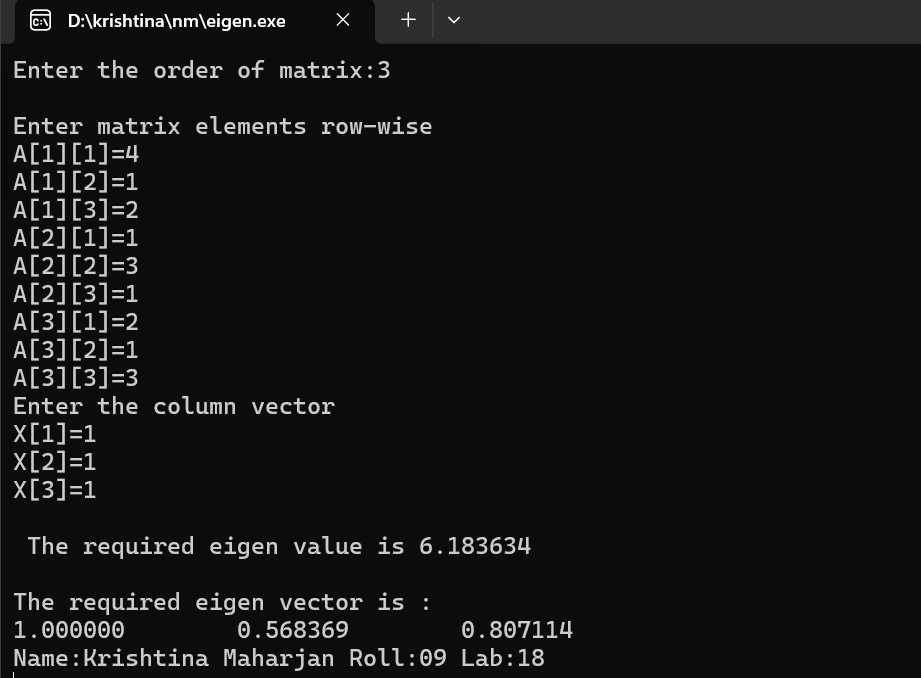
}

printf("\nName: Krishtina Maharjan Roll: 09 Lab: 18\n");

getch();

}

**Output :**



**Source Code :**

#include<stdio.h>

#include<math.h>

/\*Define the RHS of the first order differential equation here(Ex: dy/dx=f(x,y)) \*/

double f(double x, double y){

return -2\*x-y;

}

int main(){

int i;

double y,xi,yi,xf,h;

printf("Enter the initial condition for y: ");

scanf("%lf",&yi);

printf("Enter the initial condition for x: ");

scanf("%lf",&xi);

printf("Enter the value of x for which y is required: ");

scanf("%lf",&xf);

printf("Enter the step-width h: ");

scanf("%lf",&h);

printf("x\t\ty\t\ty'\t\thy'\t\ty+hy'\n");

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

//Begin Euler Routine

while(xi<xf){

y=yi+h\*f(xi,yi);

printf("%lf\t%lf\t%lf\t%lf\t%lf\n",xi,yi,f(xi,yi),h\*f(xi,yi),y);

yi=y;

xi=xi+h;

}

printf("%lf\t%lf\n",xi,yi);

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

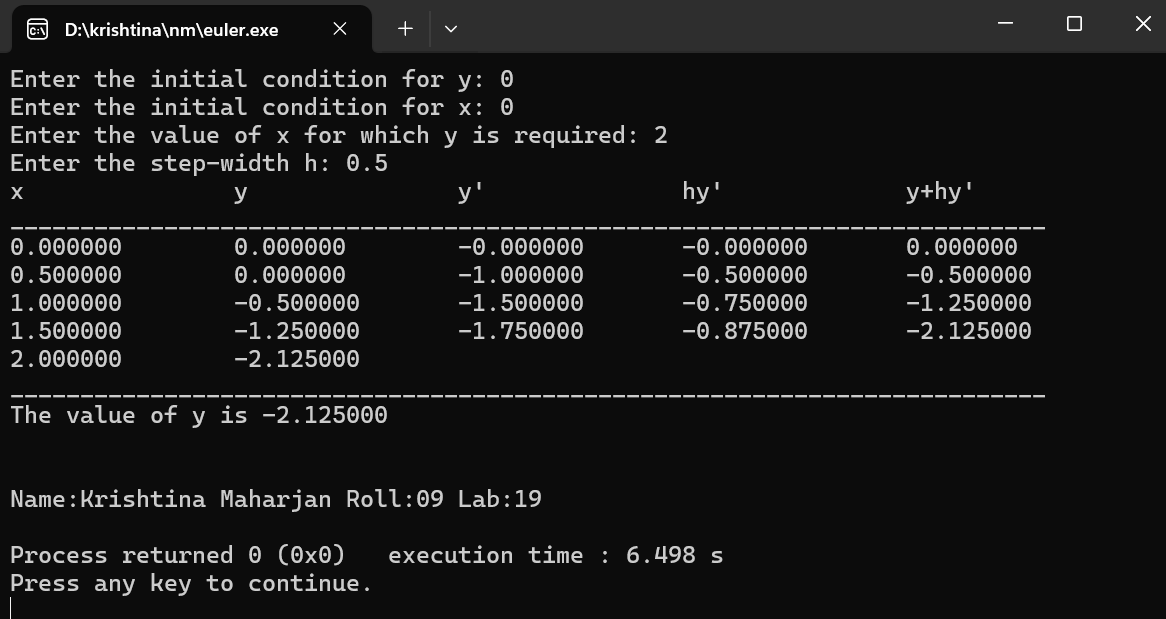
printf("The value of y is %lf\n\n",y);

printf("\nName:Krishtina Maharjan Roll:09 Lab:19\n");

return 0;

}

**Output :**



**Source Code :**

#include<stdio.h>

float f(float x){

return (6\*x+2);

}

int main(){

int i;

float xf,x,y,ye,h,k1,k2,n;

printf("Enter x and y values: ");

scanf("%f %f", &x,&y);

printf("Enter stepsize: ");

scanf("%f", &h);

printf("Enter final x value: ");

scanf("%f", &xf);

n = (xf-x)/h;

for(i=1; i<=n; i++){

k1 = f(x);

printf("k1 = %f\n", k1);

ye = y + k1 \* h;

k2 = f(x+h);

y = y+((k1+k2)/2)\*h;

x += h;

printf("%f %f\n", x,y);

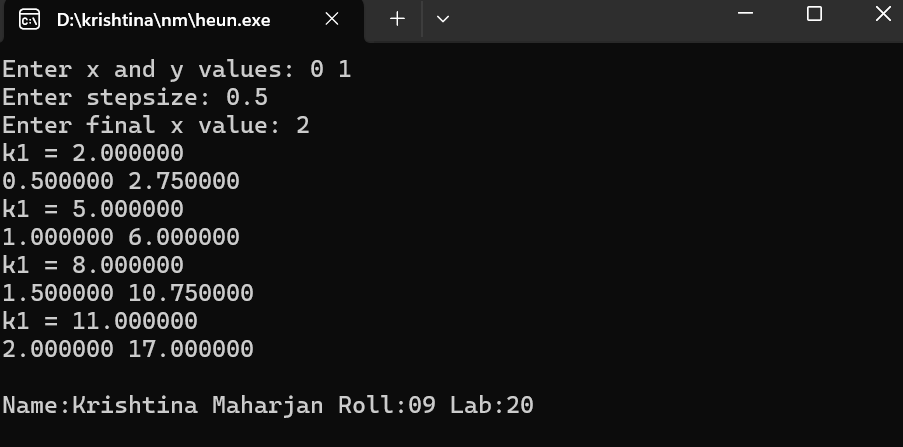
}

printf("\nName:Krishtina Maharjan Roll:09 Lab:20\n");

return 0;

}

**Output :**



**Source Code :**

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

float f1(float x,float y,float z){

return(z);

}

float f2(float x,float y,float z){

return(x+y);

}

float shoot(float x0,float y0,float z0,float xn,float h,int p){

float x,y,z,k1,k2,k3,k4,l1,l2,l3,l4,k,l,x1,y1,z1;

x=x0;

y=y0;

z=z0;

do{

k1=h\*f1(x,y,z);

l1=h\*f2(x,y,z);

k2=h\*f1(x+h/2.0,y+k1/2.0,z+l1/2.0);

l2=h\*f2(x+h/2.0,y+k1/2.0,z+l1/2.0);

k3=h\*f1(x+h/2.0,y+k2/2.0,z+l2/2.0);

l3=h\*f2(x+h/2.0,y+k2/2.0,z+l2/2.0);

k4=h\*f1(x+h,y+k3,z+l3);

l4=h\*f2(x+h,y+k3,z+l3);

l=1/6.0\*(l1+2\*l2+2\*l3+l4);

k=1/6.0\*(k1+2\*k2+2\*k3+k4);

y1=y+k;

x1=x+h;

z1=z+l;

x=x1;

y=y1;

z=z1;

if(p==1) {

printf("\n%f\t%f",x,y);

}

}while(x<xn);

return(y);

}

main(){

float x0,y0,h,xn,yn,z0,m1,m2,m3,b,b1,b2,b3,e;

int p=0;

printf(" Enter x0,y0,xn,yn,h:");

scanf("%f%f%f%f%f",&x0,&y0,&xn,&yn,&h);

printf("\n Enter the trial M1:");

scanf("%f",&m1);

b=yn;

z0=m1;

b1=shoot(x0,y0,z0,xn,h,p=1);

printf("\nB1 is %f",b1);

if(fabs(b1-b)<0.00005) {

printf("\n The value of x and respective z are:\n");

e=shoot(x0,y0,z0,xn,h,p=1);

return(0);

}

else{

printf("\nEnter the value of M2:");

scanf("%f",&m2);

z0=m2;

b2=shoot(x0,y0,z0,xn,h,p=1);

printf("\nB2 is %f",b2);

}

if(fabs(b2-b)<0.00005){

printf("\n The value of x and respective z are\n");

e= shoot(x0,y0,z0,xn,h,p=1);

return(0);

}

else{

printf("\nM2=%f\tM1=%f",m2,m1);

m3=m2+(((m2-m1)\*(b-b2))/(1.0\*(b2-b1)));

if(b1-b2==0)

exit(0);

printf("\nExact value of M =%f",m3);

z0=m3;

b3=shoot(x0,y0,z0,xn,h,p=0);

}

if(fabs(b3-b)<0.000005){

printf("\nThere is solution :\n");

e=shoot(x0,y0,z0,xn,h,p=1);

exit(0);

}

do{

m1=m2;

m2=m3;

b1=b2;

b2=b3;

m3=m2+(((m2-m1)\*(b-b2))/(1.0\*(b2-b1)));

z0=m3;

b3=shoot(x0,y0,z0,xn,h,p=0);

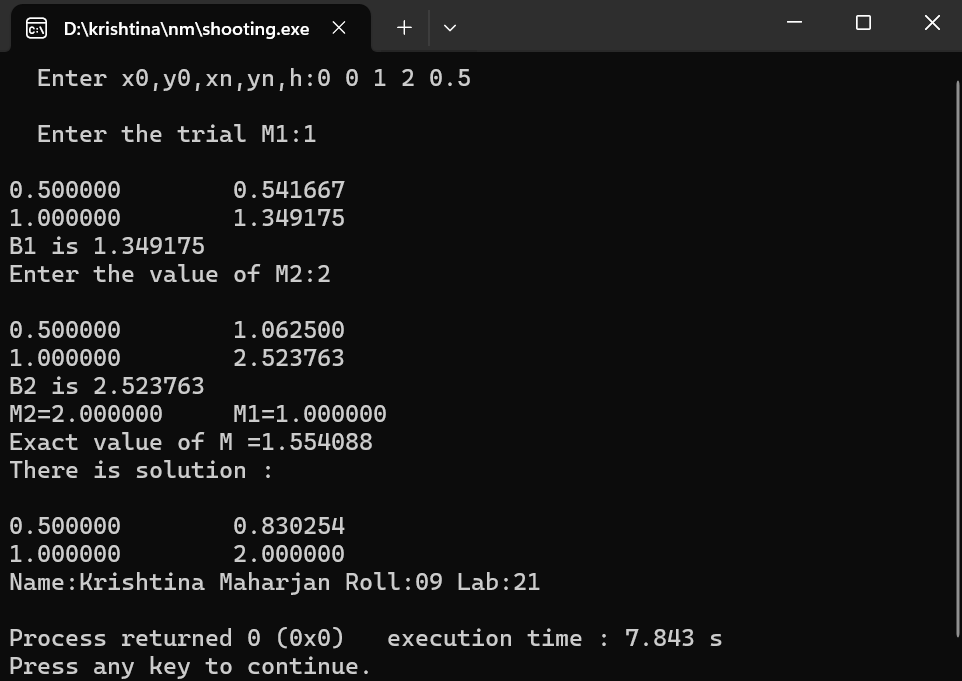
}while(fabs(b3-b)<0.0005);

z0=m3;

e=shoot(x0,y0,z0,xn,h,p=1);

}

**Output** :



**Source Code :**

#include<stdio.h>

#include<math.h>

#define S 4

typedef float newvar[S+1][S+1];

void entrow(int i, newvar u) {

int j;

printf("Enter the value of u[%d,j], j=1 to %d\n", i, S);

for(j = 1; j <= S; j++)

scanf("%f", &u[i][j]);

}

void entcol(int j, newvar u) {

int i;

printf("Enter the value of u[i,%d], i=2 to %d", j, S-1);

for(i = 2; i <= S-1; i++)

scanf("%f", &u[i][j]);

}

void oput(newvar u, int wd, int prsn) {

int i, j;

for(i = 1; i <= S; i++) {

for(j = 1; j <= S; j++)

printf("%d,%d,%f ", wd, prsn, u[i][j]);

printf("\n");

}

}

int main() {

newvar u;

float mer, ar, e, t;

int i, j, itr, maxitr;

for(i = 1; i <= S; i++) {

for(j = 1; j <= S; j++) {

u[i][j] = 0;

}

}

printf("Enter the Boundary Condition\n");

entrow(1, u);

entrow(S, u);

entcol(1, u);

entcol(S, u);

printf("Enter the allowed error and maximum number of iterations: ");

scanf("%f %d", &ar, &maxitr);

for(itr = 1; itr <= maxitr; itr++) {

mer = 0;

for(i = 2; i <= S-1; i++) {

for(j = 2; j <= S-1; j++) {

t = (u[i-1][j] + u[i+1][j] + u[i][j+1] + u[i][j-1]) / 4;

e = fabs(u[i][j] - t);

if(e > mer)

mer = e;

u[i][j] = t;

}

}

printf("Iteration Number %d\n", itr);

oput(u, 9, 2);

if(mer <= ar) {

printf("After %d iterations, the solution is:\n", itr);

oput(u, 8, 1);

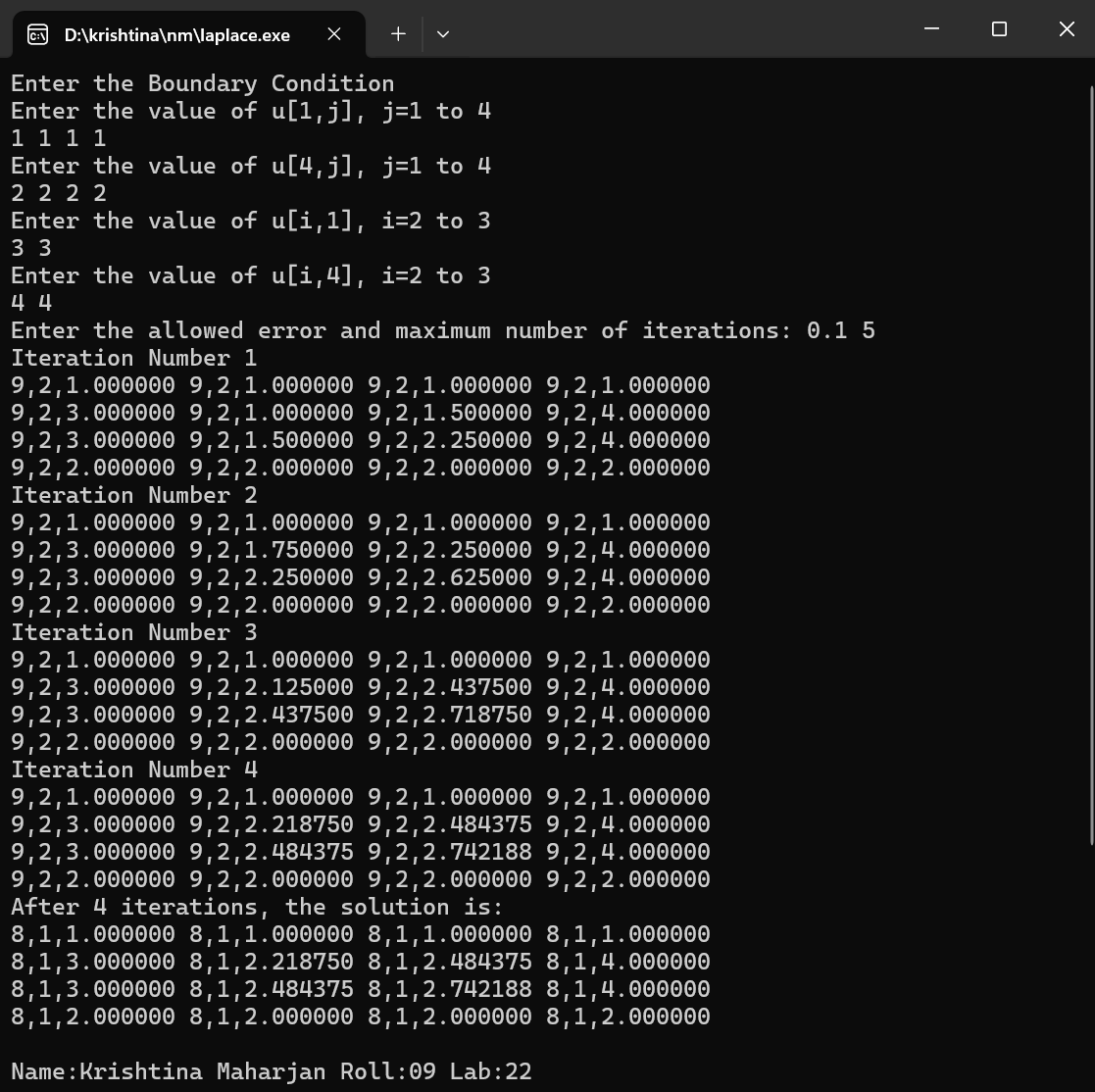
printf("\nName:Krishtina Maharjan Roll:09 Lab:22\n");

return 0;

}

}

**Output :**



**Source Code :**

#include <stdio.h>

#include <math.h>

#define g(x, y) (2 \* (x) \* (x) \* (y) \* (y))

int main() {

int n, i, j, k;

float sum, error, E[10], a[10][10], b[10], new\_x[10], old\_x[10], tl, tr, tu, tb, h;

printf("Enter Dimension of plate: ");

scanf("%d", &n);

printf("Enter Dimension of grid: ");

scanf("%f", &h);

printf("Enter temperatures at left, right, bottom & upper part of plate: ");

scanf("%f %f %f %f", &tl, &tr, &tb, &tu);

// Constructing Coefficient Matrix

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

a[i][i] = -4;

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

a[i][n-i]=0;

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

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

if (i != j && j != (n - i))

a[i][j] = 1;

}

k = 0;

for (i = 1; i < n; i++)

for (j = 1; j < n; j++)

b[k++] = h \* h \* g(i, j);

k = 0;

for (i = 1; i < n; i++) {

for (j = 1; j < n; j++) {

if ((i - 1) == 0)

b[k] -= tl;

if ((i + 1) == n)

b[k] -= tr;

if ((j - 1) == 0)

b[k] -= tb;

if ((j + 1) == n)

b[k] -= tu;

k++;

}

}

printf("Enter Accuracy Limit: ");

scanf("%f", &error);

// Solving system using Gauss-Seidel method

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

new\_x[i] = 0;

while (1) {

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

sum = b[i];

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

if (i != j)

sum -= a[i][j] \* new\_x[j];

}

old\_x[i] = new\_x[i];

new\_x[i] = sum / a[i][i];

E[i] = fabs(new\_x[i] - old\_x[i]) / fabs(new\_x[i]);

}

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

if (E[i] > error)

break;

}

if (i == (n + 1))

break;

else

continue;

}

printf("Solution:\n");

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

printf("x[%d] = %.6f\n", i + 1, new\_x[i]);

printf("\nName:Krishtina Maharjan Roll:09 Lab:23\n");

return 0;

}

**Output :**

