# Experiment 1

**AIM: To solve using Bisection Method**

#include <iostream>

#include <cmath>

using namespace std;

double function(double x) {

return (x - cos(x));

}

int main() {

double midx, function\_mid, left, right, previous\_function\_mid;

left = 0, right = 1;

midx = (left + right) / 2;

previous\_function\_mid = function(midx);

while(true) {

double value\_at\_left = function(left);

double value\_at\_right = function(right);

double value\_at\_mid = function(midx);

if(value\_at\_mid \* value\_at\_left < 0)

right = midx;

else

left = midx;

previous\_function\_mid = value\_at\_mid;

midx = (left + right) / 2;

if(abs(previous\_function\_mid) <= 0.00001)

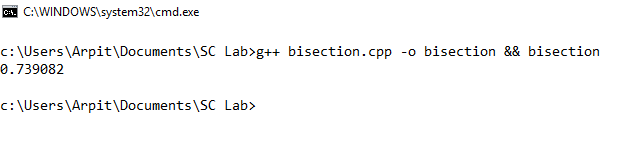
break;

}

cout << midx << "\n";

}

ANS = 0.739082



# Experiment 2

**AIM: To solve using Regula Falsi Method**.

#include <iostream>

#include <cmath>

using namespace std;

float f(float x) {

return (x - cos(x));

}

int main() {

float a = 0, b = 1;

float x = 0, y = 0;

while(true) {

x = (a \* f(b) - b \* f(a)) / (f(b) - f(a));

if(f(a) \* f(b) < 0)

b = x;

else

a = x;

if(y == x)

break;

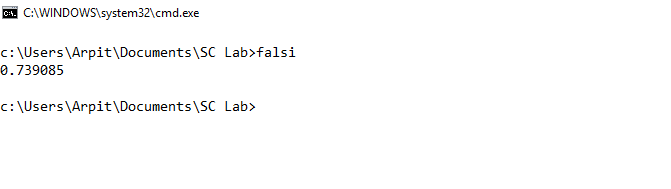
y = x;

}

cout << x << "\n";

}

ANSWER = 0.739085



# Experiment 3

**AIM: To solve using Newton Raphson Method**

#include <iostream>

#include <cmath>

using namespace std;

double function(double x) {

return (pow(x, 3) - (3) \* (x) + 1);

}

double derivative(double x) {

return (3 \* pow(x, 2) - 3);

}

int main() {

double xn = 0, xn1;

while(true) {

xn1 = xn - (function(xn) / derivative(xn));

if(xn1 == xn)

break;

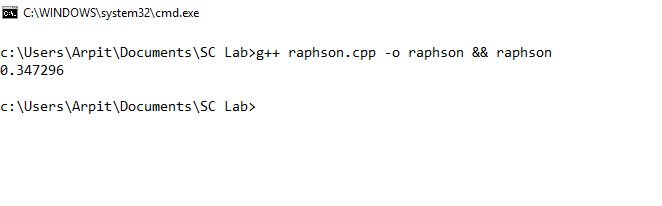
xn = xn1;

}

cout << xn1 << "\n";

}

ANSWER = 0.347296



# Experiment 4

**AIM: To solve the given system of equations using Gauss Elimination Method**

#include<iostream>

using namespace std;

void transform(float arr[4][5], int src, int dest, float factor){

for(int i = 0 ; i < 5 ; i++)

arr[dest][i] = arr[dest][i] + arr[src][i] \* factor;

}

int main(){

float arr[4][5] = {10, -7, 3, 5, 6,

-6, 8, -1, -4, 5,

3, 1, 4, 11, 2,

5, -9, -2, 4, 7};

transform(arr, 0, 1, -arr[1][0] / arr[0][0]);

transform(arr, 0, 2, -arr[2][0] / arr[0][0]);

transform(arr, 0, 3, -arr[3][0] / arr[0][0]);

transform(arr, 1, 2, -arr[2][1] / arr[1][1]);

transform(arr, 1, 3, -arr[3][1] / arr[1][1]);

transform(arr, 2, 3, -arr[3][2] / arr[2][2]);

for(int i = 0 ; i < 4 ; i++){

for(int j = 0 ; j < 5 ; j++)

cout<<arr[i][j]<< " ";

cout<<endl;

}

float fin[4];

for(int i = 3 ; i >= 0 ; i--) {

cout << "x" << (i + 1) << " : ";

float ans = arr[i][4];

for(int j = 0 ; j < 4 ; j++) {

if(j != i)

ans -= arr[i][j] \* fin[j];

}

ans /= arr[i][i];

fin[i] = ans;

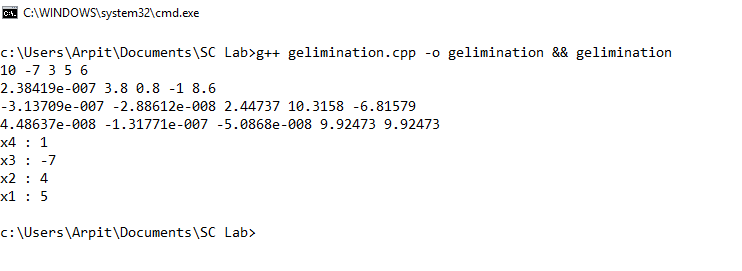
cout << ans << endl;

}

return 0;

}

ANS = (5, 4, -7, 1)



# Experiment 5

**AIM: To solve the system of equations sing Gauss Seidal Method**

#include <iostream>

#include <vector>

#include <cmath>

using namespace std;

int main() {

ios\_base::sync\_with\_stdio(0);

vector<vector<float> > matrix;

matrix.resize(3, vector<float>(4));

for(int i = 0 ; i < 3 ; i++)

for(int j = 0 ; j < 4 ; j++)

cin >> matrix[i][j];

vector<float> ans(3, 0);

bool pass = true;

while(pass) {

pass = false;

for(int i = 0 ; i < 3 ; i++) {

float val = matrix[i][3];

for(int j = 0 ; j < 3 ; j++)

if(i != j)

val -= matrix[i][j] \* ans[j];

val /= matrix[i][i];

if(abs(val - ans[i]) > 0.00001)

pass = true;

ans[i] = val;

}

}

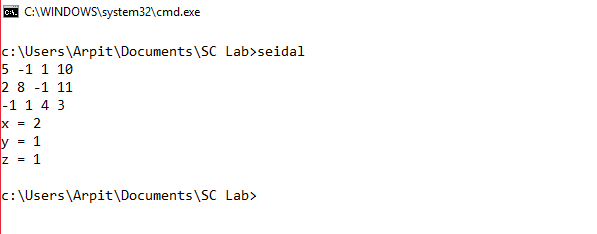
cout << "x = " << ans[0] << "\n";

cout << "y = " << ans[1] << "\n";

cout << "z = " << ans[2] << "\n";

}

ANS = (2, 1, 1)



# Experiment 6

**AIM: Use Newton Forward Interpolation Formula to get value at x = 1895.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***X*** | 1891 | 1901 | 1911 | 1921 | 1931 |
| ***Y*** | 46 | 66 | 81 | 93 | 101 |

#include <iostream>

#include <vector>

using namespace std;

typedef float f;

vector<f> y;

f p = 0.4, n = 5, ans = 0;

float factorial(float n) {

if(n <= 1)

return 1;

else

return (n \* factorial(n - 1));

}

void solve(float k) {

if(k == n)

return;

float prod = y[0];

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

prod \*= (p - i);

prod /= factorial(k);

ans += prod;

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

y[i] = y[i + 1] - y[i];

solve(k + 1);

}

int main() {

y.resize(5, 0);

y[0] = 46; y[1] = 66; y[2] = 81; y[3] = 93;

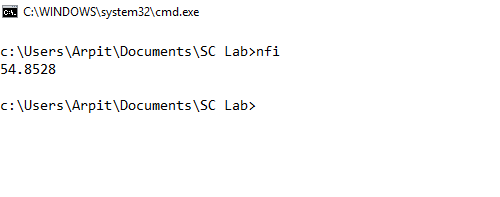
y[4] = 101;

solve(0);

cout << ans << "\n";

}

ANS = 54.8528



# Experiment 7

**AIM: Find the value of y at x = 9.5 using Lagrange’s Method**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***X*** | 7 | 8 | 9 | 10 |
| ***Y*** | 3 | 1 | 1 | 9 |

#include <iostream>

using namespace std;

float point = 9.5;

int x[] = {7, 8, 9, 10};

int y[] = {3, 1, 1, 9};

int main() {

float answer = 0;

for(int index = 0 ; index < 4 ; index++) {

float term = y[index];

for(int i = 0 ; i < 4 ; i++)

if(index != i)

term \*= (point - x[i]) / (x[index] - x[i]);

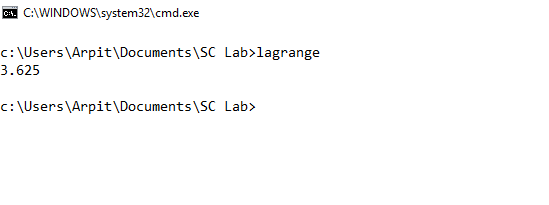
answer += term;

}

cout << answer << "\n";

}

ANS = 3.625



# Experiment 8

**AIM: Evaluate the integral using Simpson’s 1/3rd Formula**

#include <iostream>

#include <cmath>

using namespace std;

float h = 0.2, lower = 4, upper = 5.2;

double f(double x) {

return log(x);

}

int main() {

ios\_base::sync\_with\_stdio(0);

double x = lower + h, sum = 0;

while(x < upper) {

double temp = f(x);

sum += temp;

x += h;

cout << sum << " " << x << "\n";

}

sum \*= 2;

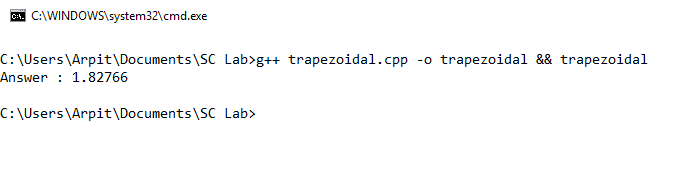
sum += (f(lower) + f(upper));

sum \*= (h / 2);

cout << "Answer : " << sum << "\n";

}

ANSWER : 1.82766



# Experiment 9

**AIM: Evaluate the integral using Simpson’s 1/3rd Formula**

#include <iostream>

#include <cmath>

using namespace std;

double h = 0.2, lower = 4, upper = 5.2;

double f(double x) {

return log(x);

}

int main() {

double x = lower + h, sum = 0;

int counter = 1;

while(x < upper) {

if(counter % 2 == 0)

sum += (2 \* f(x));

else

sum += (4 \* f(x));

counter++;

x += h;

}

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

sum \*= (h / 3);

cout << sum << “\n”;

}

ANSWER : 1.82785

