

**Source code:**

#include<stdio.h>

#include<cmath>

#include<iomanip>

using namespace std;

int fact(int x) {

if(x == 1 || x == 0) {

return 1;

}

return x \* fact(x - 1);

}

double poisson\_prob(double lambda, int x) {

return (exp(-lambda) \* pow(lambda, x)) / fact(x);

}

void poisson(int lambda) {

double temp;

for(int i = 0; i < 15; i++) {

temp = poisson\_prob(lambda, i);

cout << "x = " << i << ": "<<fixed << setprecision(4)<<temp << endl;

}

}

int main() {

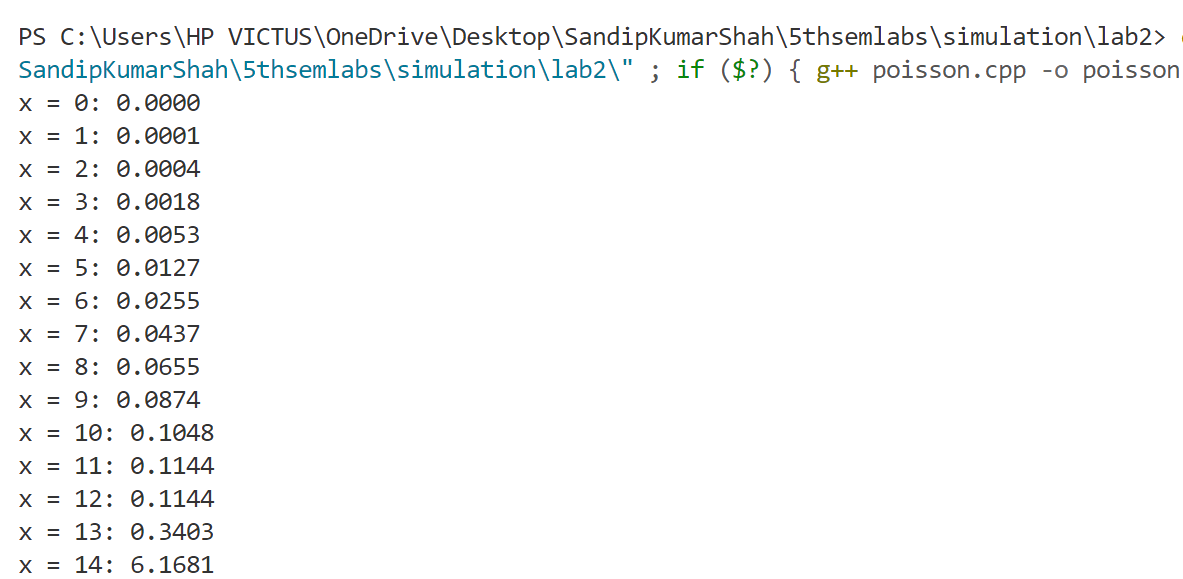
int lambda = 12;

poisson(lambda);

return 0;

}

**Output:**



**2. Write a program to estimate the value of PI using Monte Carlo Simulation.**

**Source Code:**

#include <iostream>

#include <cstdlib> // For rand()

#include <ctime> // For time()

using namespace std;

double estimatePi(int points) {

int insideCircle = 0;

double x, y;

srand(time(0));

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

x = (double)rand() / RAND\_MAX;

y = (double)rand() / RAND\_MAX;

if(x \* x + y \* y <= 1) {

insideCircle++;

}

}

return 4.0 \* insideCircle / points;

}

int main() {

int points;

cout << "Enter number of points to simulate: ";

cin >> points;

// Input validation

if (points <= 0) {

cout << "ERROR! Number of points must be positive." << endl;

return 1;

}

if (points < 1000) {

cout << "WARNING: Using fewer than 1000 points may give inaccurate results." << endl;

}

double pi = estimatePi(points);

cout << "Estimated value of PI: " << pi << endl;

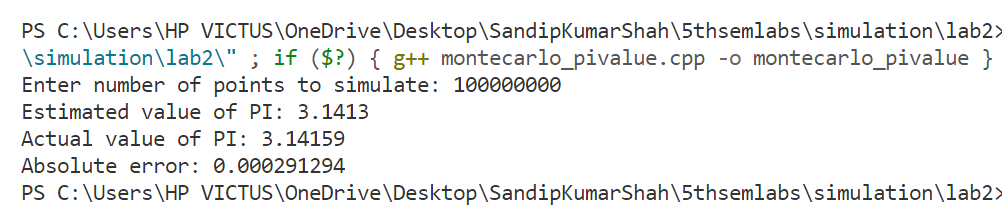
cout << "Actual value of PI: " << 3.14159265359 << endl;

cout << "Absolute error: " << abs(pi - 3.14159265359) << endl;

return 0;

}

**Output:**



**3. Write a program to estimate the area under the curve using the Monte Carlo Simulation.**

**Source Code:**

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <cmath>

using namespace std;

double f(double x) {

return x \* x;

}

double monteCarloArea(double a, double b, double maxHeight, int points) {

int underCurve = 0;

double x, y;

srand(time(0));

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

x = a + (b - a) \* ((double)rand() / RAND\_MAX);

y = maxHeight \* ((double)rand() / RAND\_MAX);

if(y <= f(x)) {

underCurve++;

}

}

double rectangleArea = (b - a) \* maxHeight;

return rectangleArea \* underCurve / points;

}

int main() {

int points;

double a = 0.0;

double b = 1.0;

double maxHeight = 1.0;

cout << "Enter number of points to simulate: ";

cin >> points;

double area = monteCarloArea(a, b, maxHeight, points);

cout << "Estimated area under x^2 from " << a << " to " << b << ": " << area << endl;

cout << "Actual area (1/3 for x^2 from 0 to 1): " << 1.0/3.0 << endl;

cout << "Absolute error: " << abs(area - 1.0/3.0) << endl;

return 0;

}

**Output:**

