**CMPSC 201 Lab Report**

**CMPSC LAB 2**

**974343924 - Ibrahim Ibrahim, Youssef El-Minwai**

**November 2, 2018**

**Pseudocode**

Gather the following inputs from the user:

* The number of intervals.
* The left and right interval values.

Complete the following tasks:

* You will create loops for all four of the quadrature techniques (Left, Right, Midpoint, and Trapezoid)
* Use the math function mentioned above, ***f(x) = 1/x + 3 sin (x) - 2*** inside each of the quadrature techniques.
* Output the results of each quadrature technique.
* Calculate and out the Simpson approximation, ***(1/3) \* (2\*(Midpoint approximation) + (Trapezoid approximation))***

**Code**

#include<iostream>

#include<math.h>

using namespace std;

int main() {

cout << "This program was written to help the user solve the definite integral, f(x)= 1/x + 3\*sin(x)-2" << endl;

double intervals;

cout << "Enter the number of interval " << endl;

cin >> intervals;

double rightvalue;

cout << "Enter the right interval value " << endl;

cin >> rightvalue;

double leftvalue;

cout << "Enter the left interval value " << endl;

cin >> leftvalue;

double deltaX;

deltaX = (rightvalue - leftvalue) / intervals;

cout << "The width of the rectangles is " << deltaX << endl;

double Left\_Reimann;

double Right\_Reimann;

double Midpoint\_Reimann;

double TrapozidRule2;

double Simpsons\_Rule;

double f1;

f1 = 0;

double f2;

f2 = 0;

double f3;

f3 = 0;

double f4;

f4 = 0;

double RIGHT;

RIGHT = 0;

double LEFT;

LEFT = 0;

double MIDDLE;

MIDDLE = 0;

double TRAP;

TRAP = 0;

for (double D = leftvalue; D <= rightvalue - deltaX; D = D + deltaX)

{

f1 = 1 / D + 3 \* sin(D) - 2;

LEFT += f1;

}

Left\_Reimann = deltaX \* LEFT;

//Right\_Reimann

for (double D = leftvalue + deltaX; D <= rightvalue; D += deltaX)

{

f2 = (1 / D) + 3 \* sin(D) - 2;

RIGHT += f2;

}

Right\_Reimann = deltaX \* RIGHT;

//Midpoint\_Reimann

for (double D = leftvalue; D <= rightvalue - deltaX; D += deltaX)

{

double middle;

middle = (.5)\*(2 \* D + deltaX);

f3 = (1 / middle) + 3 \* sin(middle) - 2;

MIDDLE += f3;

}

Midpoint\_Reimann = deltaX \* MIDDLE;

//Trapozid\_Rule

for (double D = leftvalue + deltaX; D <= (rightvalue - deltaX); D += deltaX)

{

f4 = (1 / D) + 3 \* sin(D) - 2;

TRAP += f4;

}

double A1;

A1 = (1 / leftvalue) + 3 \* sin(leftvalue) - 2;

double A2;

A2 = (1 / rightvalue) + 3 \* sin(rightvalue) - 2;

TRAP = A1 + A2 + (2 \* TRAP);

TrapozidRule2 = .5\*deltaX\*TRAP;

//Simpsons\_Rule

Simpsons\_Rule = ((2 \* Midpoint\_Reimann) + TrapozidRule2)\*(1.0 / 3);

cout << "The approximation of the given integral using a Right Reimann Sum is " << Right\_Reimann << endl;

cout << "The approximation of the given integral using a Left Reimann Sum is " << Left\_Reimann << endl;

cout << "The approximation of the given integral using a Midpoint Reimann Sum is " << Midpoint\_Reimann << endl;

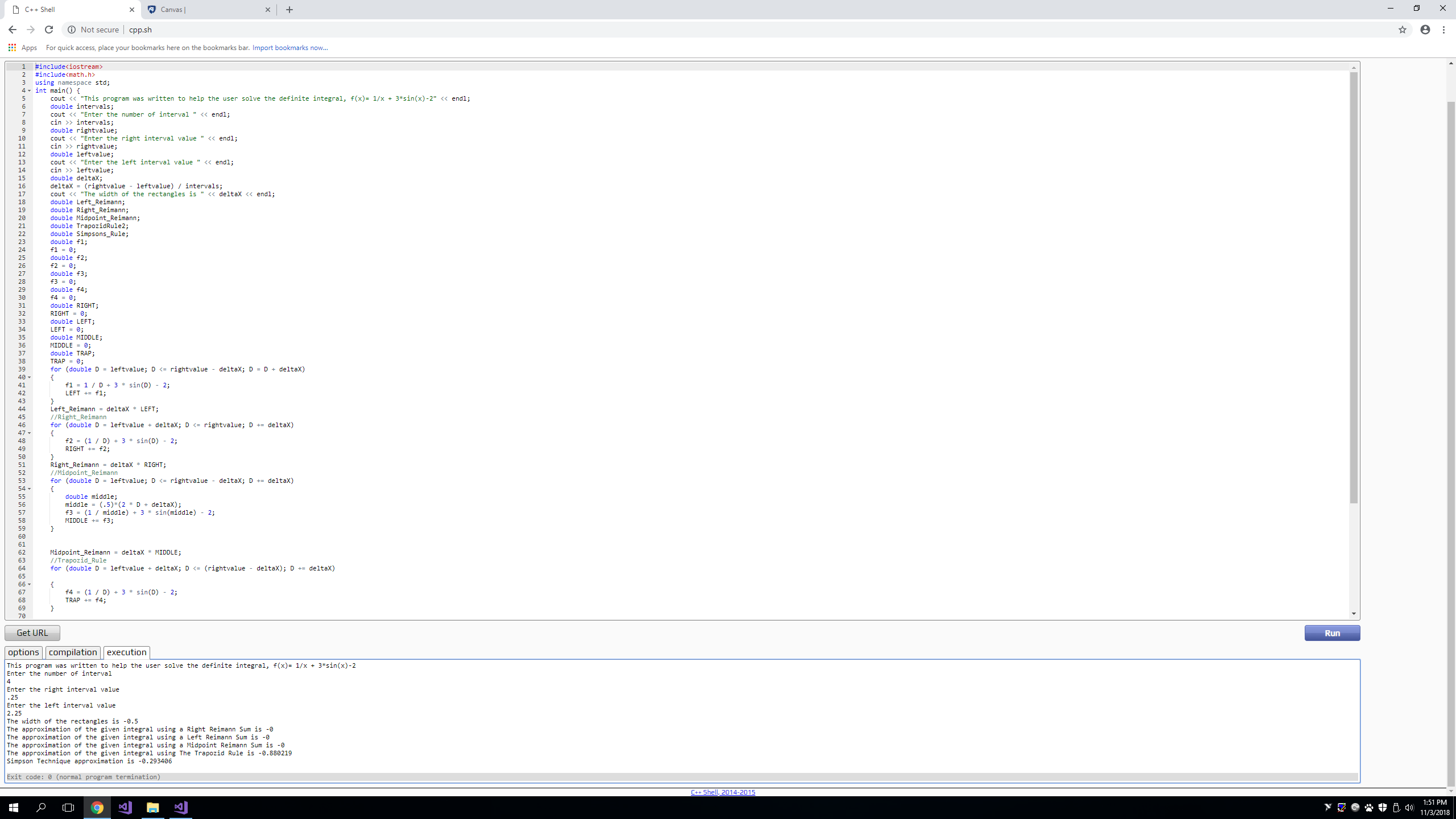
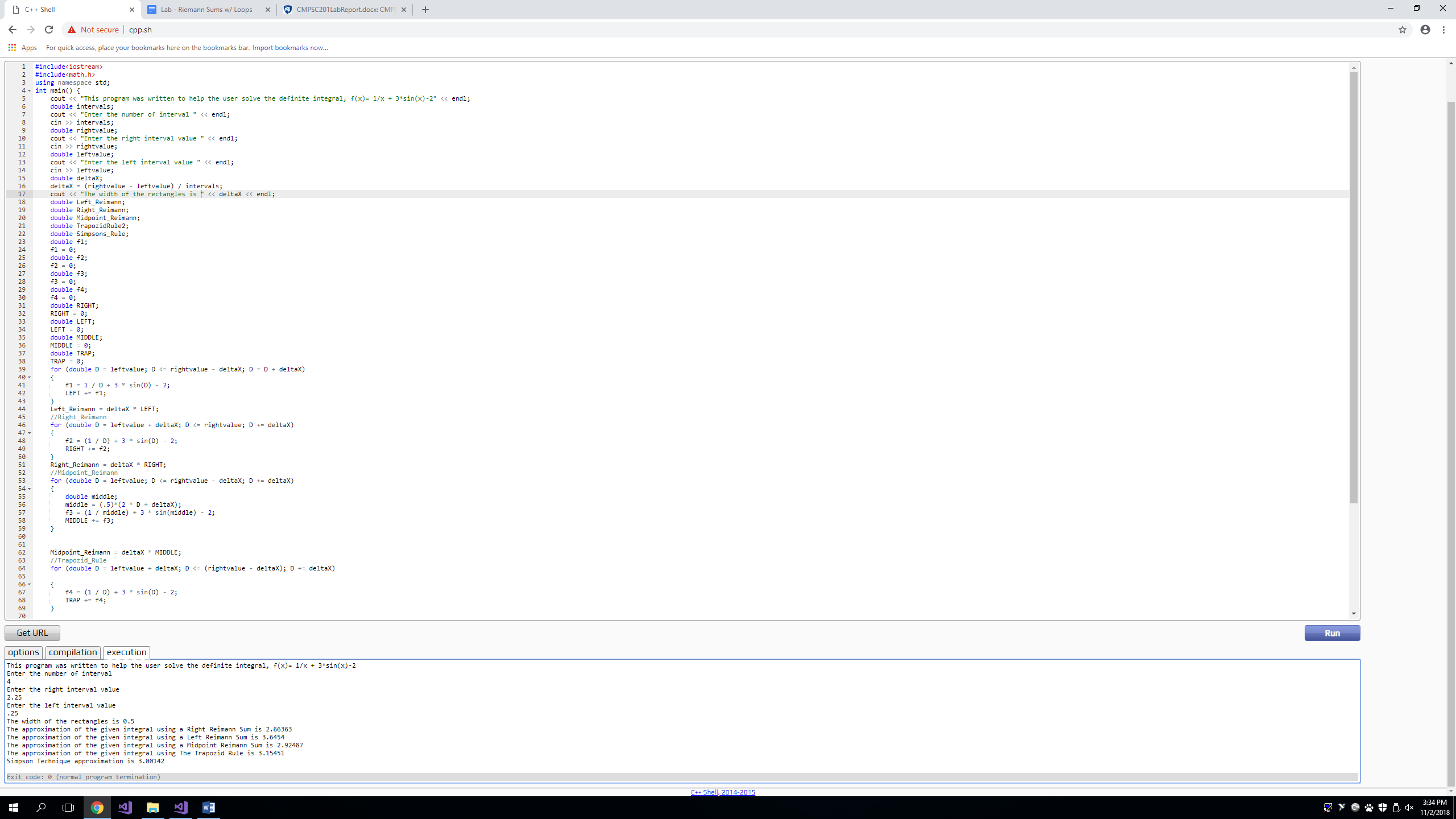
cout << "The approximation of the given integral using The Trapozid Rule is " << TrapozidRule2 << endl;

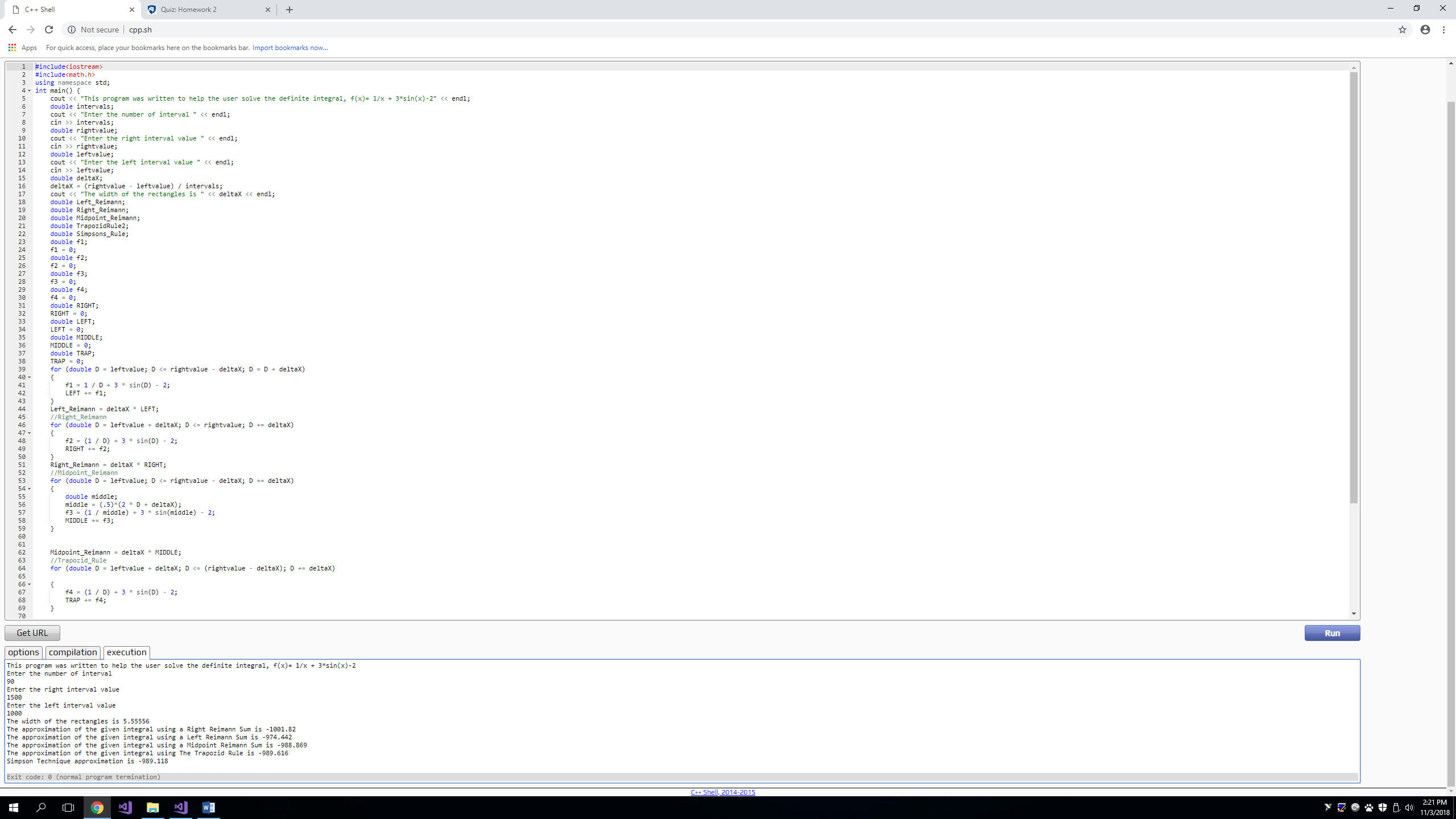
cout << "Simpson Technique approximation is " << Simpsons\_Rule << endl;

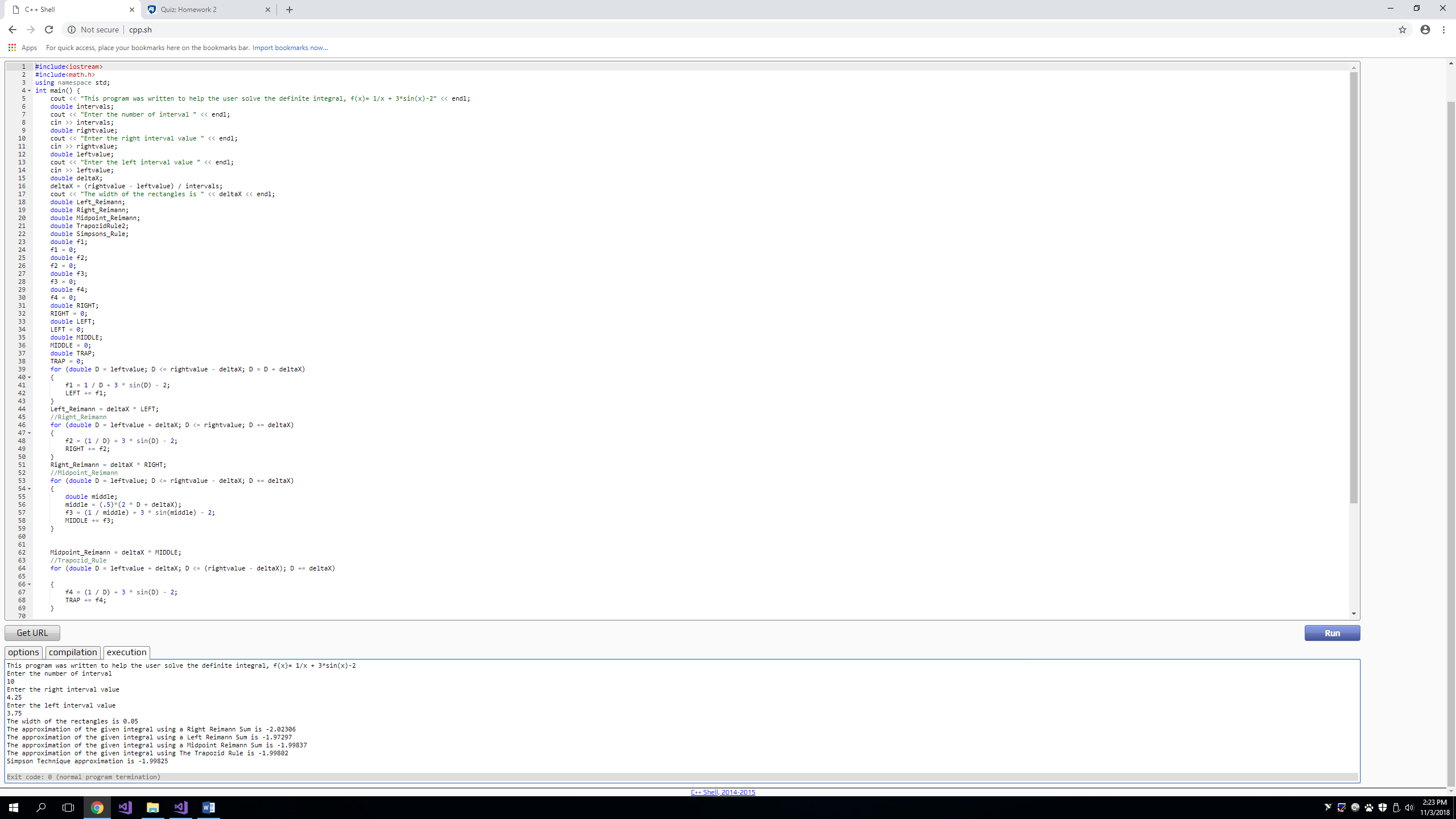
return 0;

}

**Sample Runs**







**Discussion**

1. **How many hours did this program take?**

This program has taken our group almost two weeks from start to finish, approximately 8 hours in total to finish all the work.

1. **What type of problems did you run into while programming this Project? How did you resolve them?**

One of the biggest problems we ran into during this project is determining the height of the intervals for each Riemann technique and the Trapezoid Rule. Also determining the type of loop used for the four above techniques was difficult to choose as each loop type had certain pros and cons on how it could be controlled. The first problem was figured out by introducing a variable, known as D in the code, for each technique that starts at a specific value depending on the technique and increase by the length of each interval which was set in the code as deltaX, then the height of each interval was determined by using that variable set for each technique by plugging it into the function being integrated as a definite integral. The choice of the loop became the “for” loop as it made more sense since it allowed the coder to set different conditions for the loop to run properly in the code. The variable D was set to three conditions in each technique and the “for” loop offered that without the hassle of making sure to terminate a loop so it would not continue to infinity and beyond.

1. **What was the hardest part of this Lab?**

The hardest part of this lab was introducing the concept of a graphing grid into the code or for a lack of a better explanation, allowing the code to understand that it is working with different heights based on where the starting point is based on the techniques of approximations being used. Using loops for such matter made it easy for the code to understand where to start for each technique and the conditions for each loop allowed for the calculations of the height of each interval for the four approximation techniques using the given functions being integrated as a definite integral with set boundaries on an interval.