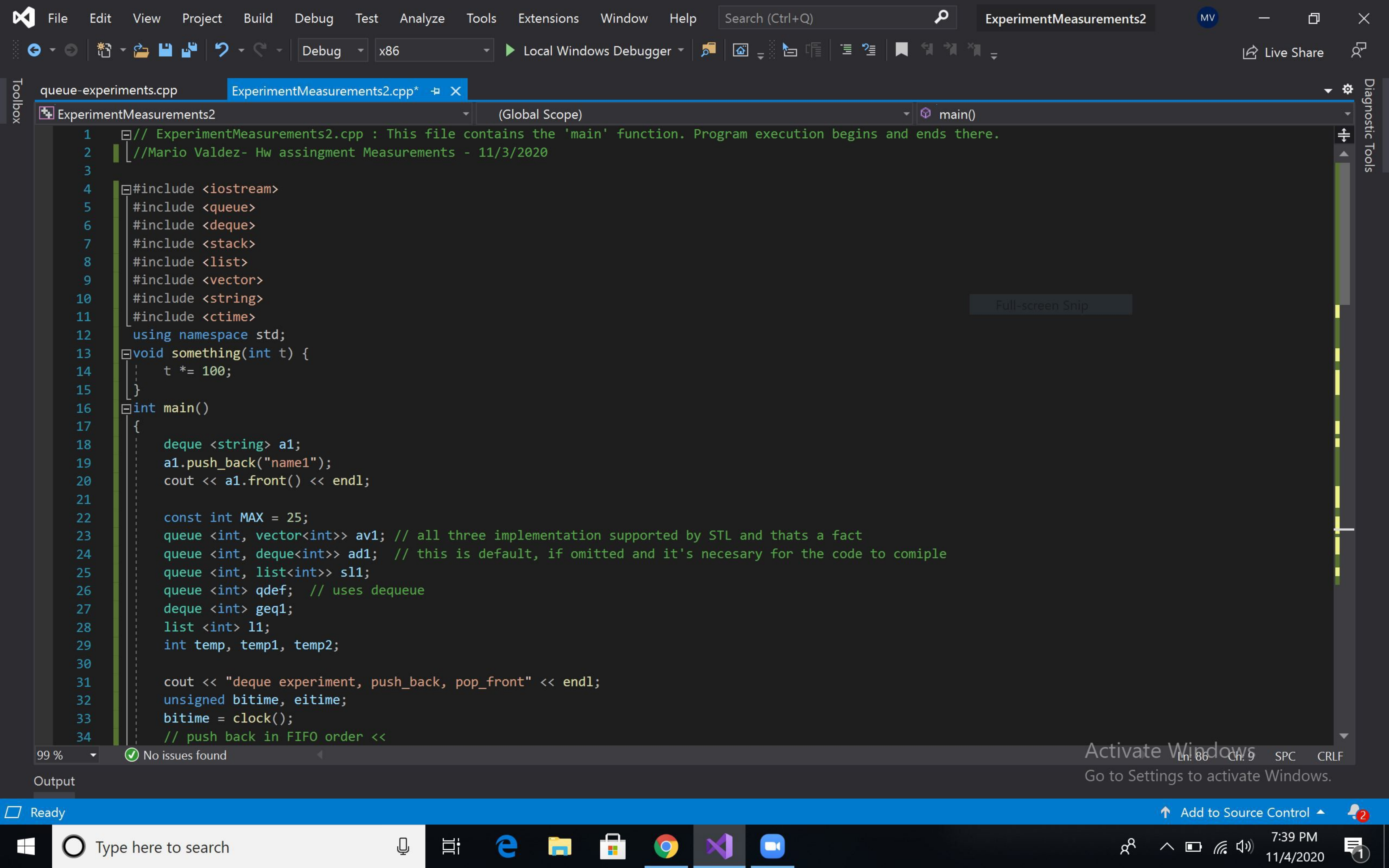
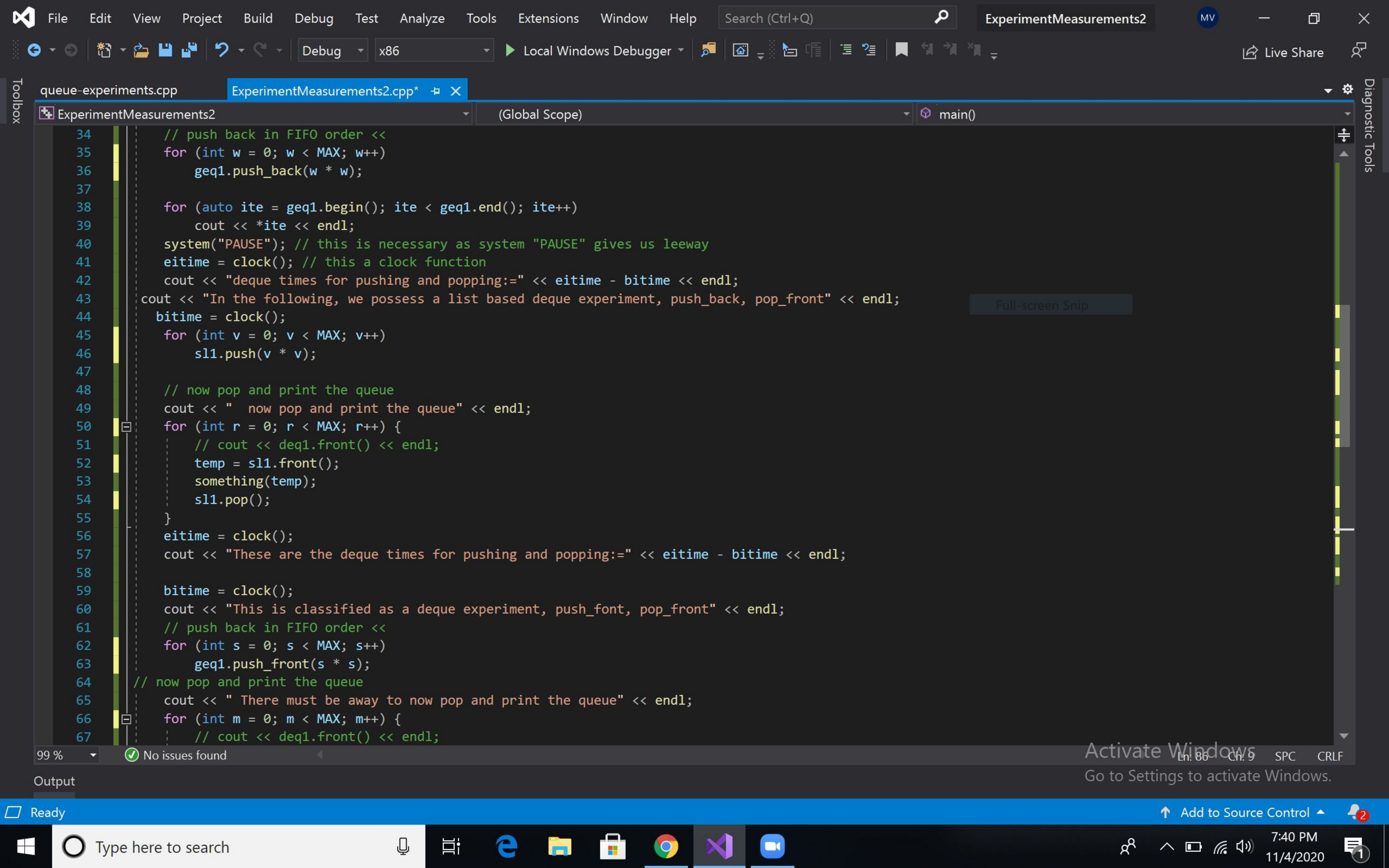
Introduction: For this special project, I have to implement at least three different kinds of queues and measure the operations for push() and pop(). I will explain some of the most complex data structures and summarize the performance numbers I received whilst compiling my code. It should be measured in terms of microseconds per operation.

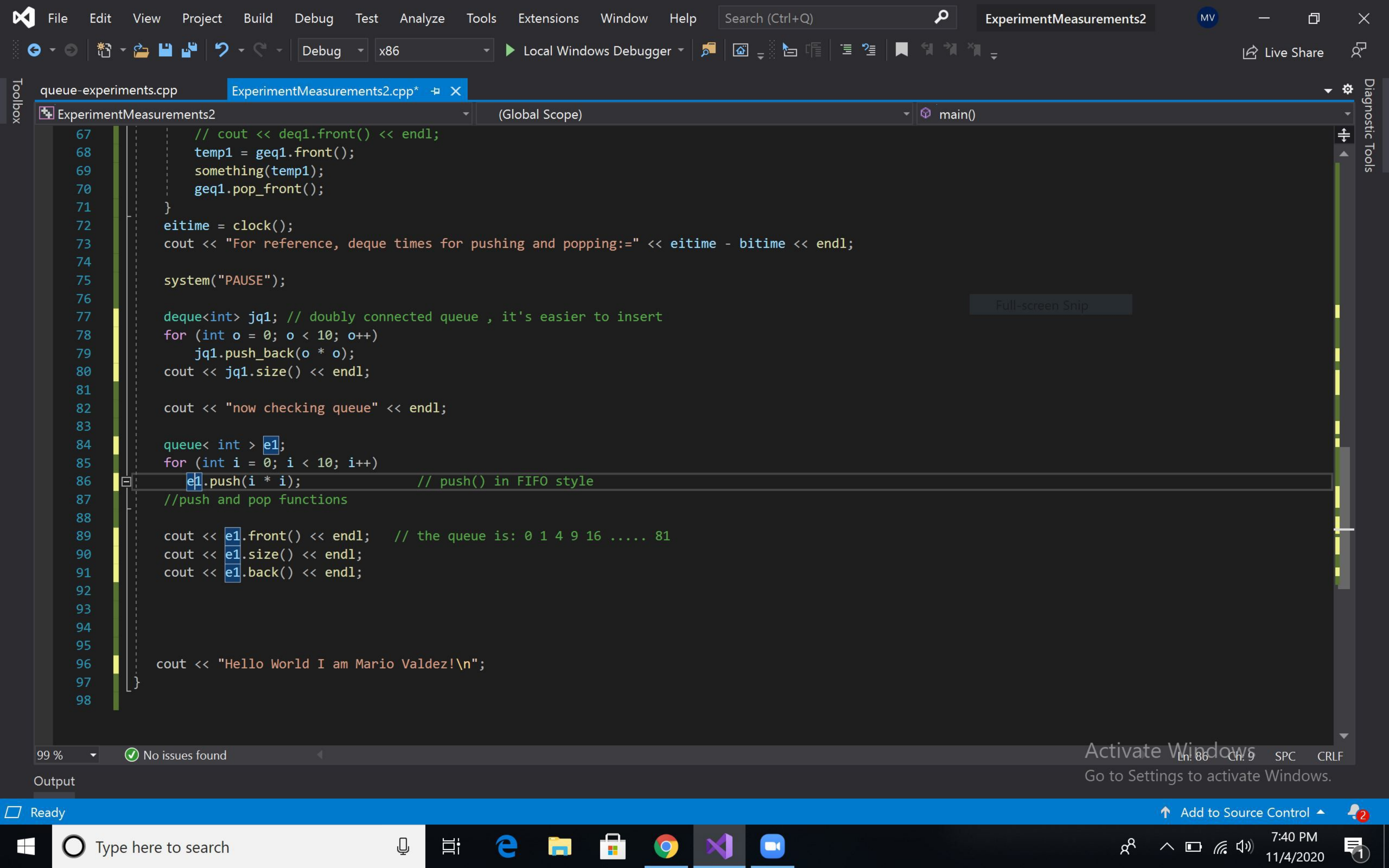
Detailed Analysis and Summary: My work contains a lot of relatively fascinating concepts that are put to the test. There was ample usage of stacks, queues, and deques present in my subsequent code. Before moving, I should make note that deques stand for double-ended queues The best way to describe a deque or double-ended queue is that it is a class in which an ordered collection with a front and back contains elements which can be efficiently added and removed from both the front and back. This is why it seamlessly manages to put together the many advantages of a stack as well as a queue. And queues are different themselves from deques. For example, a queue is associated with FIFO and stack LIFO which is a stack that is used a lot by the major compilers, to store function return addresses. FIFO tends to stand for first-in-first-out, and LIFO stands for last in first out. They feature quite heavily in code lines 22-29, and I call them by using queue <int, vector<int>> av1; (and so forth). This is necessary as all 3 implementations are supported by STL so it's a big factor in queue experiment measurements. Of course, this will make use of the measuring operations for both push() and pop(). A big example of the push function is present in code line 38, and it is listed as "geq1.push_back(w * w);", which managed to aid my code superbly.



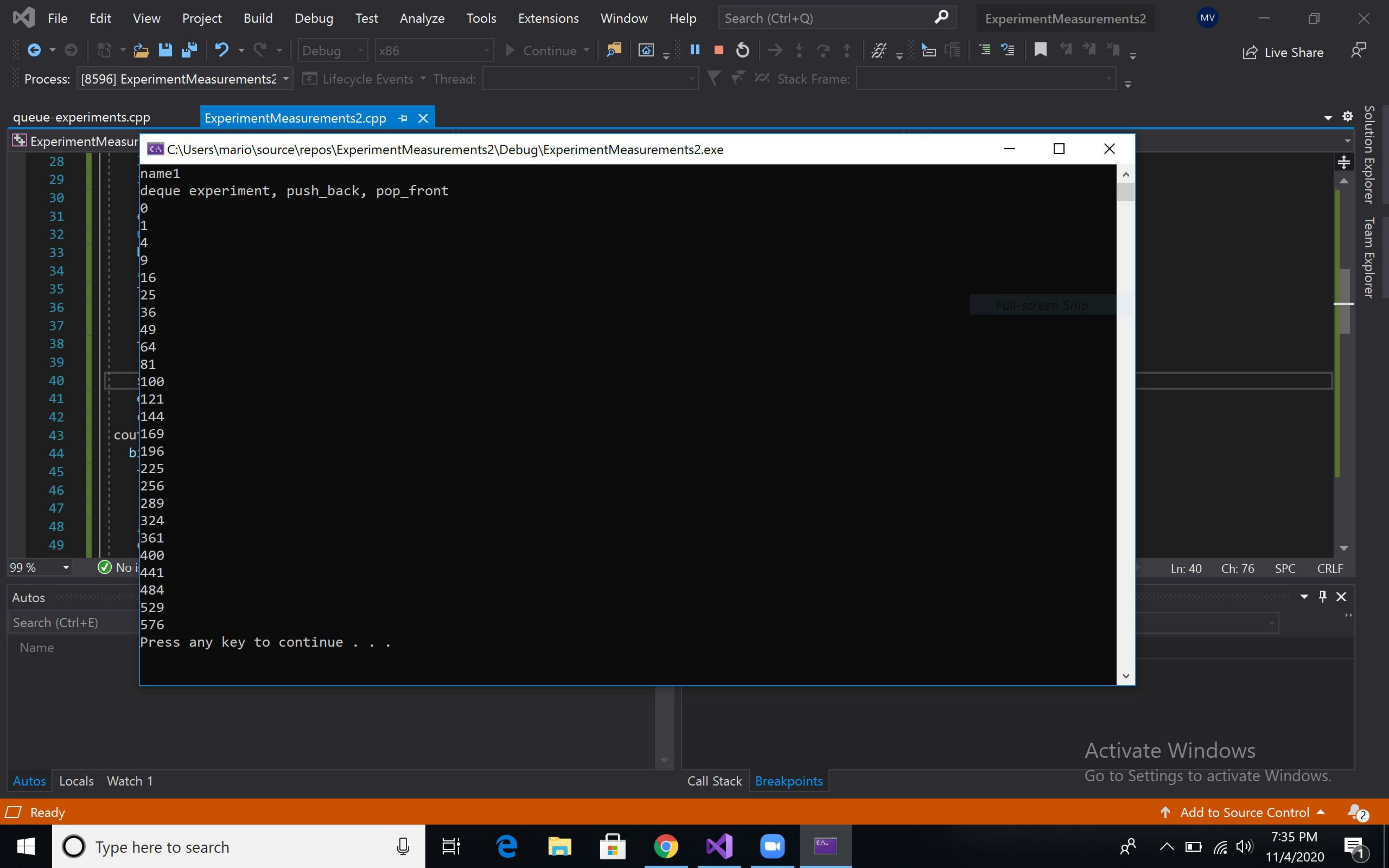
```cpp
// ExperimentMeasurements2.cpp : This file contains the 'main' function. Program execution begins and ends there.
//Mario Valdez- Hw assingment Measurements - 11/3/2020

#include <iostream>
#include <queue>
#include <deque>
#include <stack>
#include <list>
#include <vector>
#include <string>
#include <ctime>
using namespace std;
void something(int t) {
    t *= 100;
}
int main()
{
    deque <string> a1;
    a1.push_back("name1");
    cout << a1.front() << endl;

    const int MAX = 25;
    queue <int, vector<int>> av1; // all three implementation supported by STL and thats a fact
    queue <int, deque<int>> ad1;  // this is default, if omitted and it's necesary for the code to comiple
    queue <int, list<int>> sl1;
    queue <int> qdef;  // uses dequeue
    deque <int> geq1;
    list <int> l1;
    int temp, temp1, temp2;

    cout << "deque experiment, push_back, pop_front" << endl;
    unsigned bitime, eitime;
    bitime = clock();
    // push back in FIFO order <<
```



```cpp
    // push back in FIFO order <<
    for (int w = 0; w < MAX; w++)
        geq1.push_back(w * w);

    for (auto ite = geq1.begin(); ite < geq1.end(); ite++)
        cout << *ite << endl;
    system("PAUSE"); // this is necessary as system "PAUSE" gives us leeway
    eitime = clock(); // this a clock function
    cout << "deque times for pushing and popping:=" << eitime - bitime << endl;
 cout << "In the following, we possess a list based deque experiment, push_back, pop_front" << endl;
   bitime = clock();
    for (int v = 0; v < MAX; v++)
        sl1.push(v * v);

    // now pop and print the queue
    cout << "  now pop and print the queue" << endl;
    for (int r = 0; r < MAX; r++) {
        // cout << deq1.front() << endl;
        temp = sl1.front();
        something(temp);
        sl1.pop();
    }
    eitime = clock();
    cout << "These are the deque times for pushing and popping:=" << eitime - bitime << endl;

    bitime = clock();
    cout << "This is classified as a deque experiment, push_font, pop_front" << endl;
    // push back in FIFO order <<
    for (int s = 0; s < MAX; s++)
        geq1.push_front(s * s);
// now pop and print the queue
    cout << " There must be away to now pop and print the queue" << endl;
    for (int m = 0; m < MAX; m++) {
        // cout << deq1.front() << endl;
```



```
        // cout << deq1.front() << endl;
        temp1 = geq1.front();
        something(temp1);
        geq1.pop_front();
    }
    eitime = clock();
    cout << "For reference, deque times for pushing and popping:=" << eitime - bitime << endl;

    system("PAUSE");

    deque<int> jq1; // doubly connected queue , it's easier to insert
    for (int o = 0; o < 10; o++)
        jq1.push_back(o * o);
    cout << jq1.size() << endl;

    cout << "now checking queue" << endl;

    queue< int > e1;
    for (int i = 0; i < 10; i++)
       e1.push(i * i);               // push() in FIFO style
    //push and pop functions

    cout << e1.front() << endl;   // the queue is: 0 1 4 9 16 ..... 81
    cout << e1.size() << endl;
    cout << e1.back() << endl;




   cout << "Hello World I am Mario Valdez!\n";
}

```



C:\Users\mario\source\repos\ExperimentMeasurements2\Debug\ExperimentMeasurements2.exe

```
name1
deque experiment, push_back, pop_front
0
1
4
9
16
25
36
49
64
81
100
121
144
169
196
225
256
289
324
361
400
441
484
529
576
Press any key to continue . . .
```