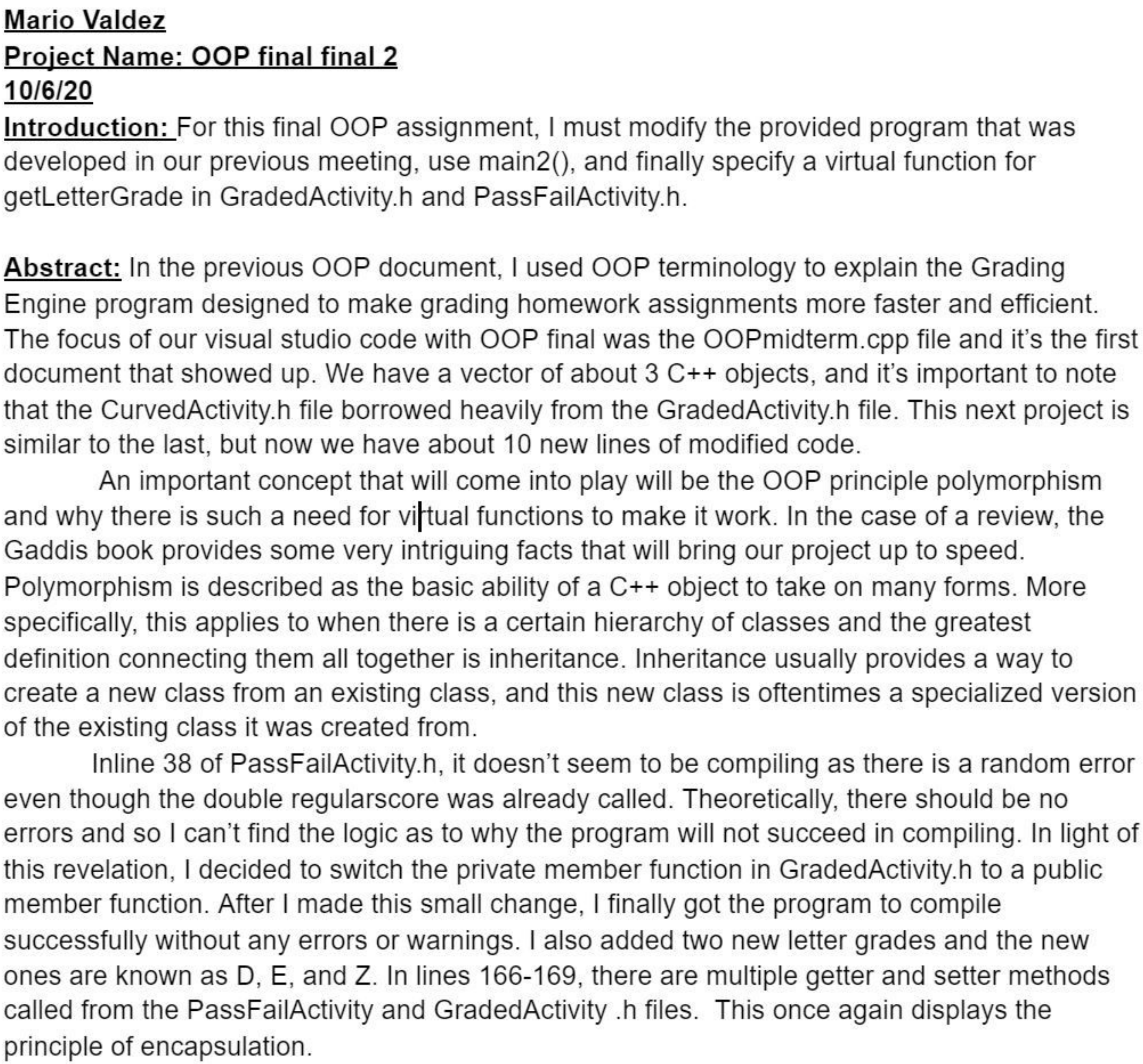


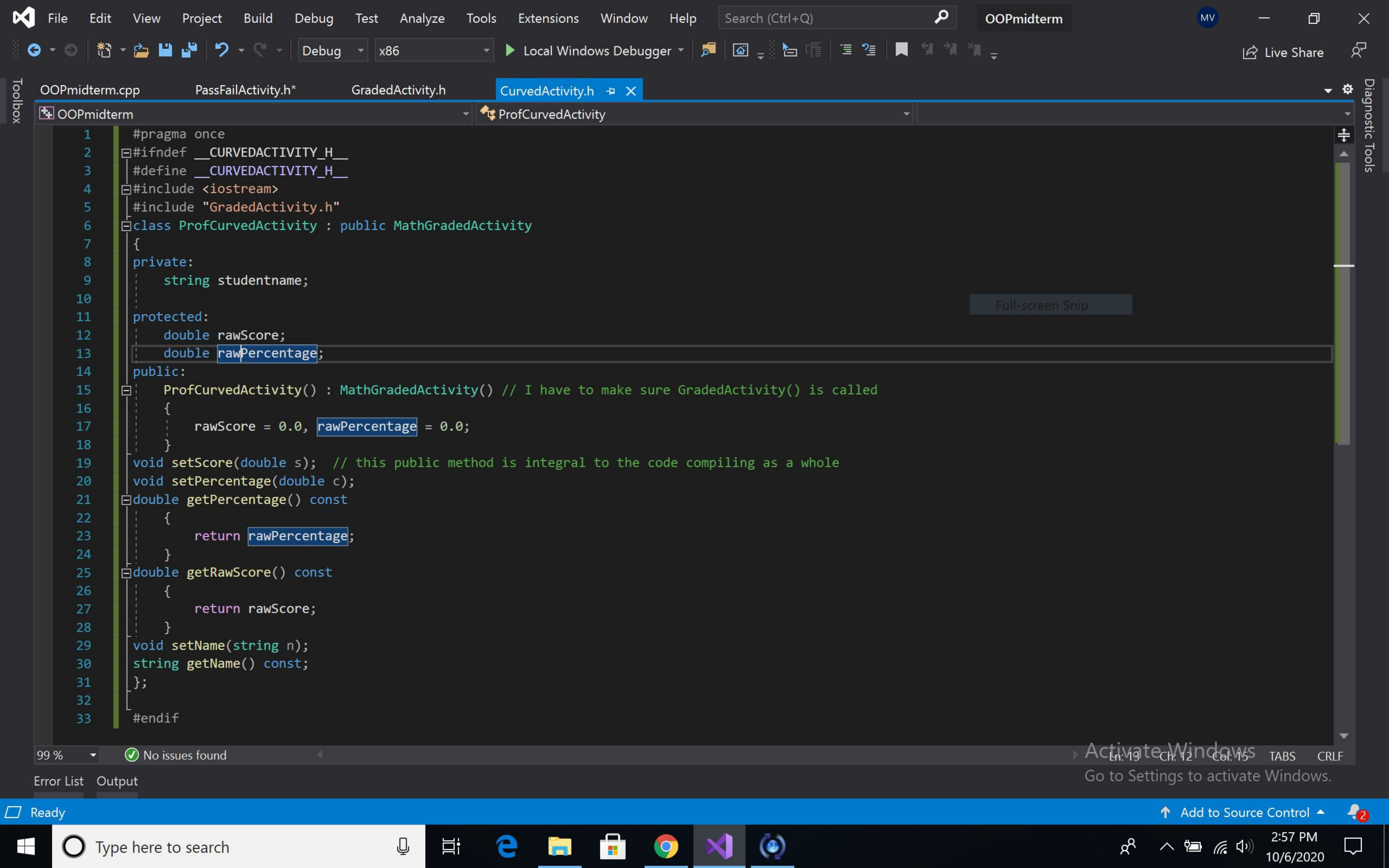
Mario Valdez
Project Name: OOP final final 2
10/6/20

Introduction: For this final OOP assignment, I must modify the provided program that was developed in our previous meeting, use main2(), and finally specify a virtual function for getLetterGrade in GradedActivity.h and PassFailActivity.h.

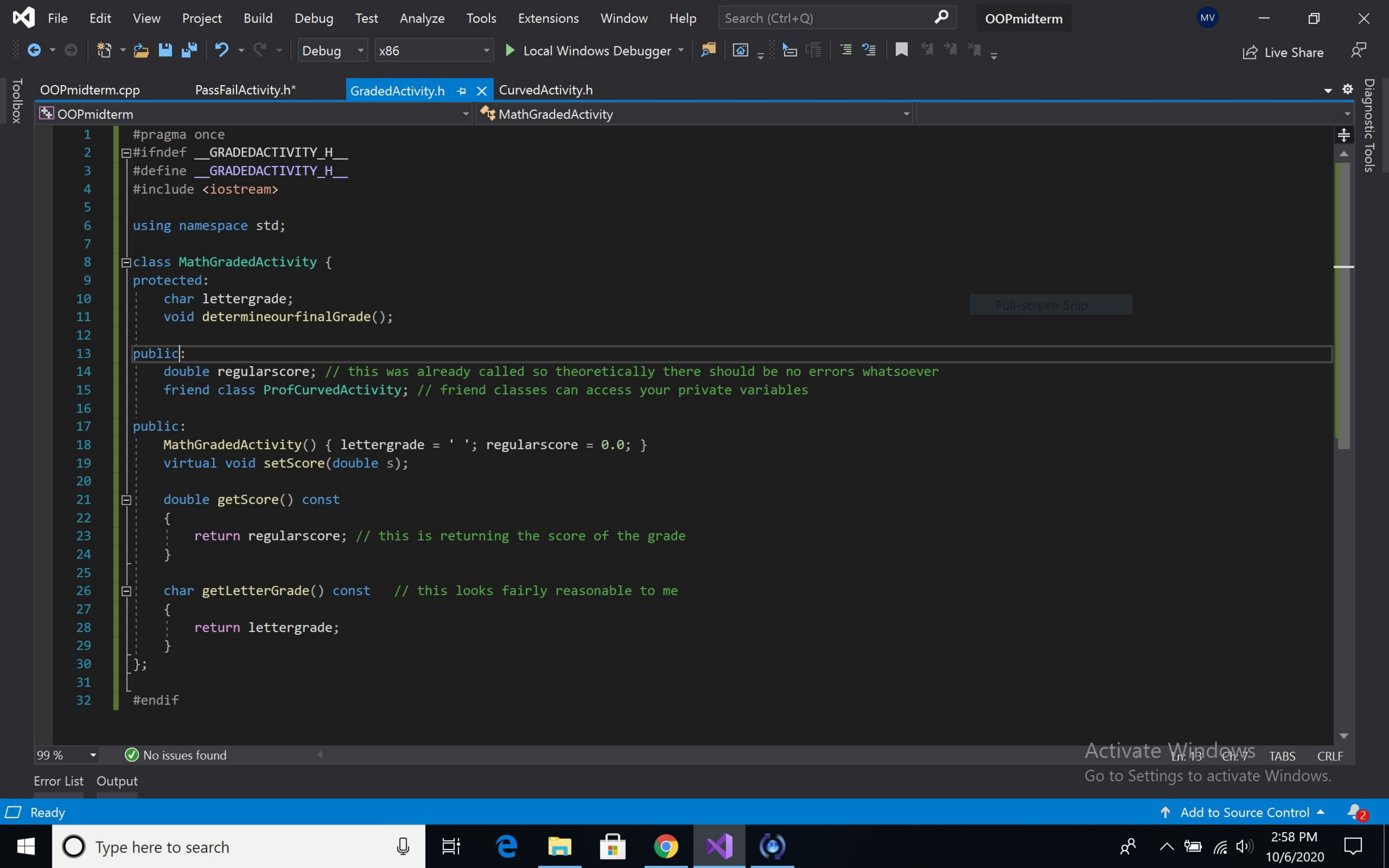
Abstract: In the previous OOP document, I used OOP terminology to explain the Grading Engine program designed to make grading homework assignments more faster and efficient. The focus of our visual studio code with OOP final was the OOPmidterm.cpp file and it's the first document that showed up. We have a vector of about 3 C++ objects, and it's important to note that the CurvedActivity.h file borrowed heavily from the GradedActivity.h file. This next project is similar to the last, but now we have about 10 new lines of modified code.

An important concept that will come into play will be the OOP principle polymorphism and why there is such a need for virtual functions to make it work. In the case of a review, the Gaddis book provides some very intriguing facts that will bring our project up to speed. Polymorphism is described as the basic ability of a C++ object to take on many forms. More specifically, this applies to when there is a certain hierarchy of classes and the greatest definition connecting them all together is inheritance. Inheritance usually provides a way to create a new class from an existing class, and this new class is oftentimes a specialized version of the existing class it was created from.

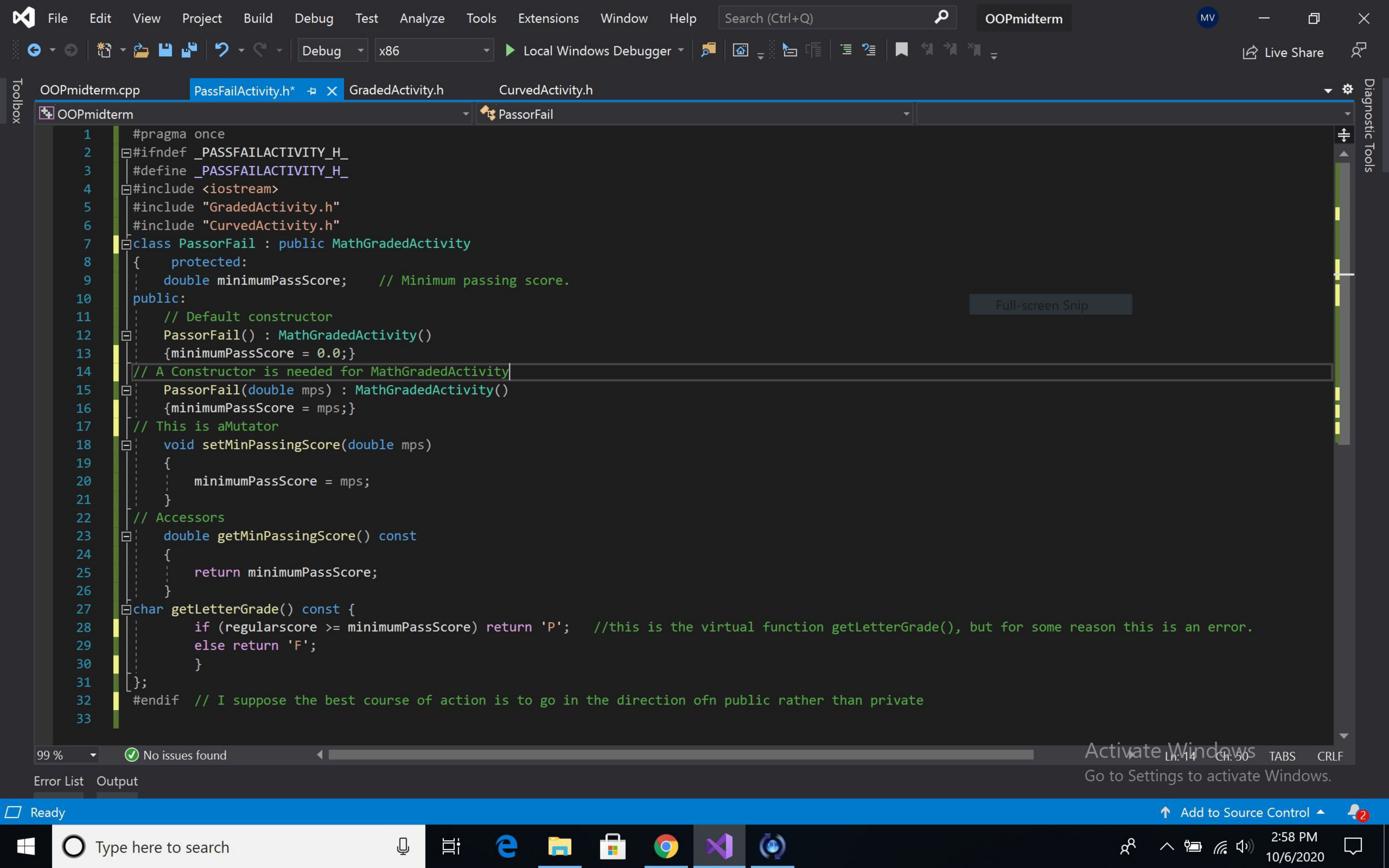
Inline 38 of PassFailActivity.h, it doesn't seem to be compiling as there is a random error even though the double regularscore was already called. Theoretically, there should be no errors and so I can't find the logic as to why the program will not succeed in compiling. In light of this revelation, I decided to switch the private member function in GradedActivity.h to a public member function. After I made this small change, I finally got the program to compile successfully without any errors or warnings. I also added two new letter grades and the new ones are known as D, E, and Z. In lines 166-169, there are multiple getter and setter methods called from the PassFailActivity and GradedActivity .h files. This once again displays the principle of encapsulation.



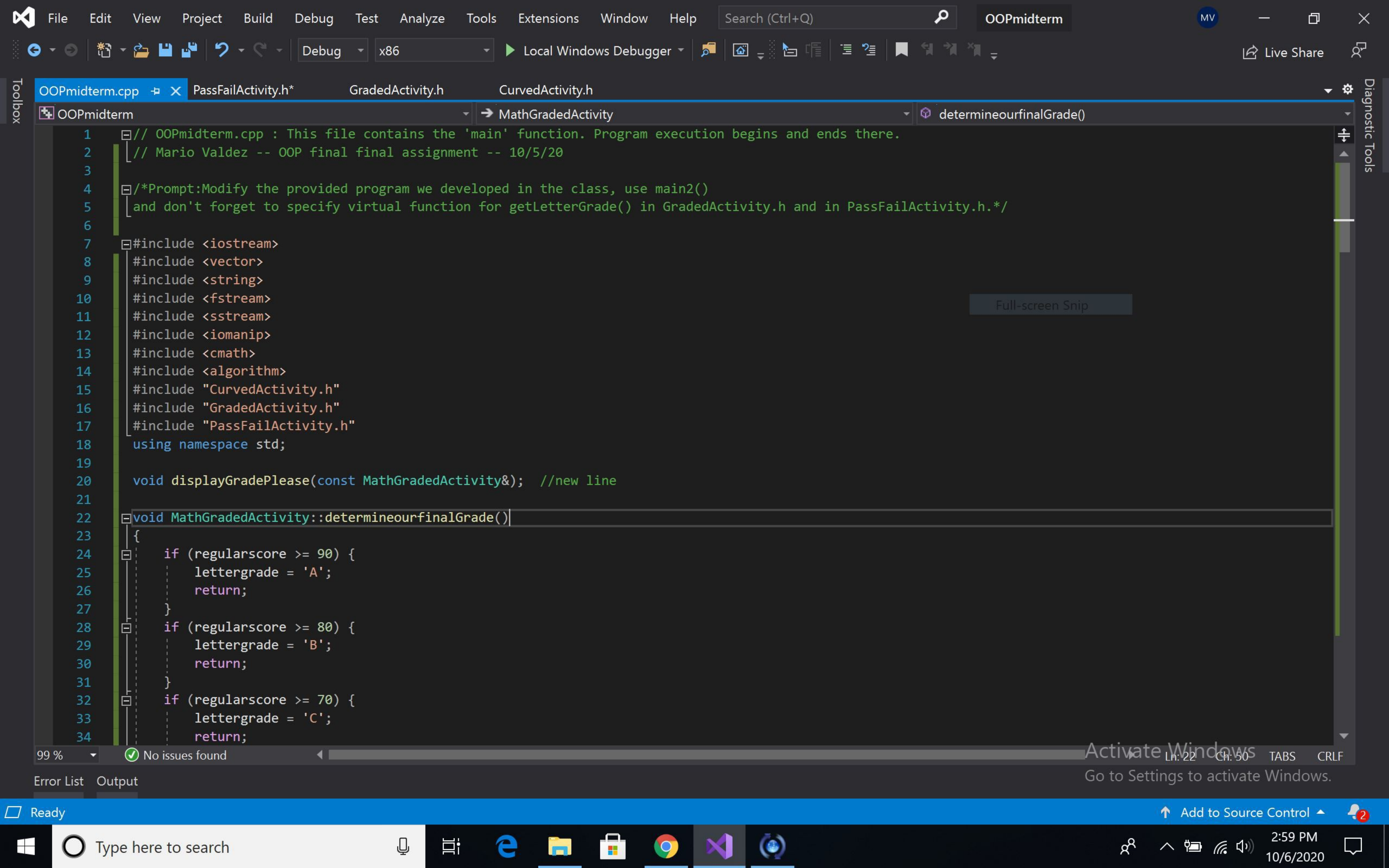
```
#pragma once
#ifndef __CURVEDACTIVITY_H__
#define __CURVEDACTIVITY_H__
#include <iostream>
#include "GradedActivity.h"
class ProfCurvedActivity : public MathGradedActivity
{
private:
    string studentname;

protected:
    double rawScore;
    double rawPercentage;
public:
    ProfCurvedActivity() : MathGradedActivity() // I have to make sure GradedActivity() is called
    {
        rawScore = 0.0, rawPercentage = 0.0;
    }
void setScore(double s);  // this public method is integral to the code compiling as a whole
void setPercentage(double c);
double getPercentage() const
    {
        return rawPercentage;
    }
double getRawScore() const
    {
        return rawScore;
    }
void setName(string n);
string getName() const;
};

#endif
```



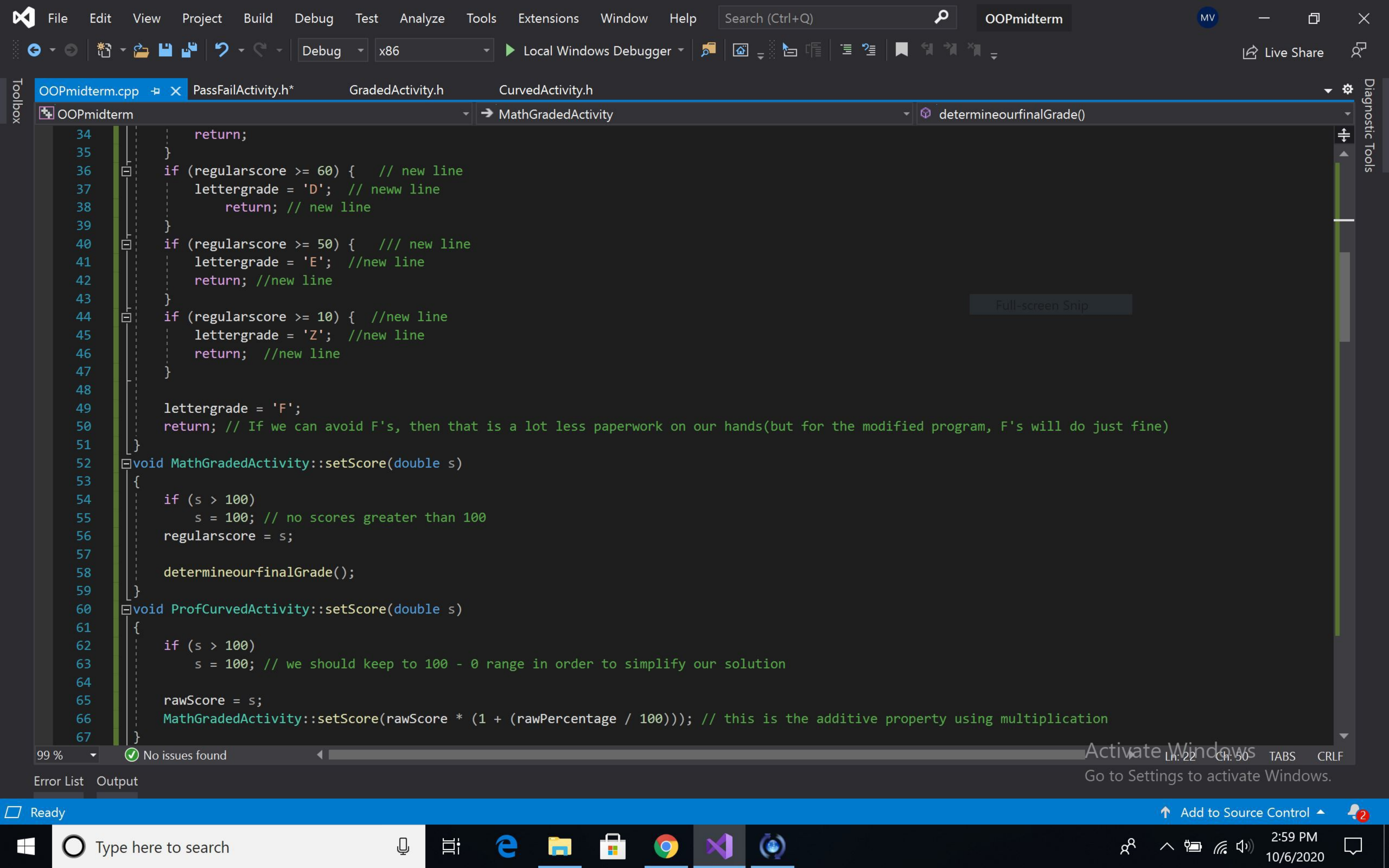
```cpp
#pragma once
#ifndef __GRADEACTIVITY_H__
#define __GRADEACTIVITY_H__
#include <iostream>

using namespace std;

class MathGradedActivity {
protected:
    char lettergrade;
    void determineourfinalGrade();

public:
    double regularscore; // this was already called so theoretically there should be no errors whatsoever
    friend class ProfCurvedActivity; // friend classes can access your private variables

public:
    MathGradedActivity() { lettergrade = ' '; regularscore = 0.0; }
    virtual void setScore(double s);

    double getScore() const
    {
        return regularscore; // this is returning the score of the grade
    }

    char getLetterGrade() const   // this looks fairly reasonable to me
    {
        return lettergrade;
    }
};

#endif
```



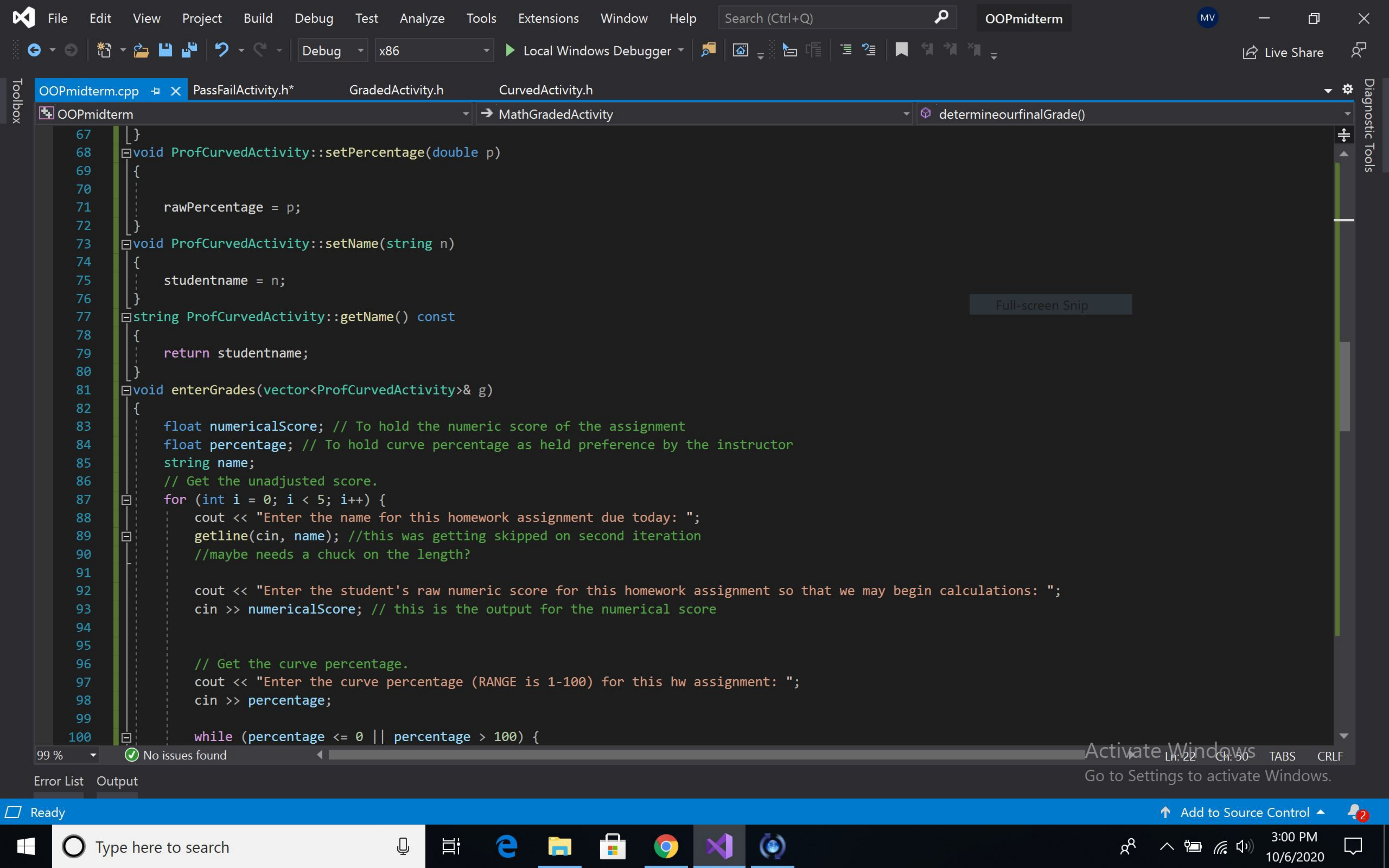
```
#pragma once
#ifndef _PASSFAILACTIVITY_H_
#define _PASSFAILACTIVITY_H_
#include <iostream>
#include "GradedActivity.h"
#include "CurvedActivity.h"
class PassorFail : public MathGradedActivity
{    protected:
    double minimumPassScore;    // Minimum passing score.
public:
    // Default constructor
    PassorFail() : MathGradedActivity()
    {minimumPassScore = 0.0;}
// A Constructor is needed for MathGradedActivity
    PassorFail(double mps) : MathGradedActivity()
    {minimumPassScore = mps;}
// This is aMutator
    void setMinPassingScore(double mps)
    {
        minimumPassScore = mps;
    }
// Accessors
    double getMinPassingScore() const
    {
        return minimumPassScore;
    }
char getLetterGrade() const {
        if (regularscore >= minimumPassScore) return 'P';   //this is the virtual function getLetterGrade(), but for some reason this is an error.
        else return 'F';
        }
};
#endif  // I suppose the best course of action is to go in the direction ofn public rather than private
```



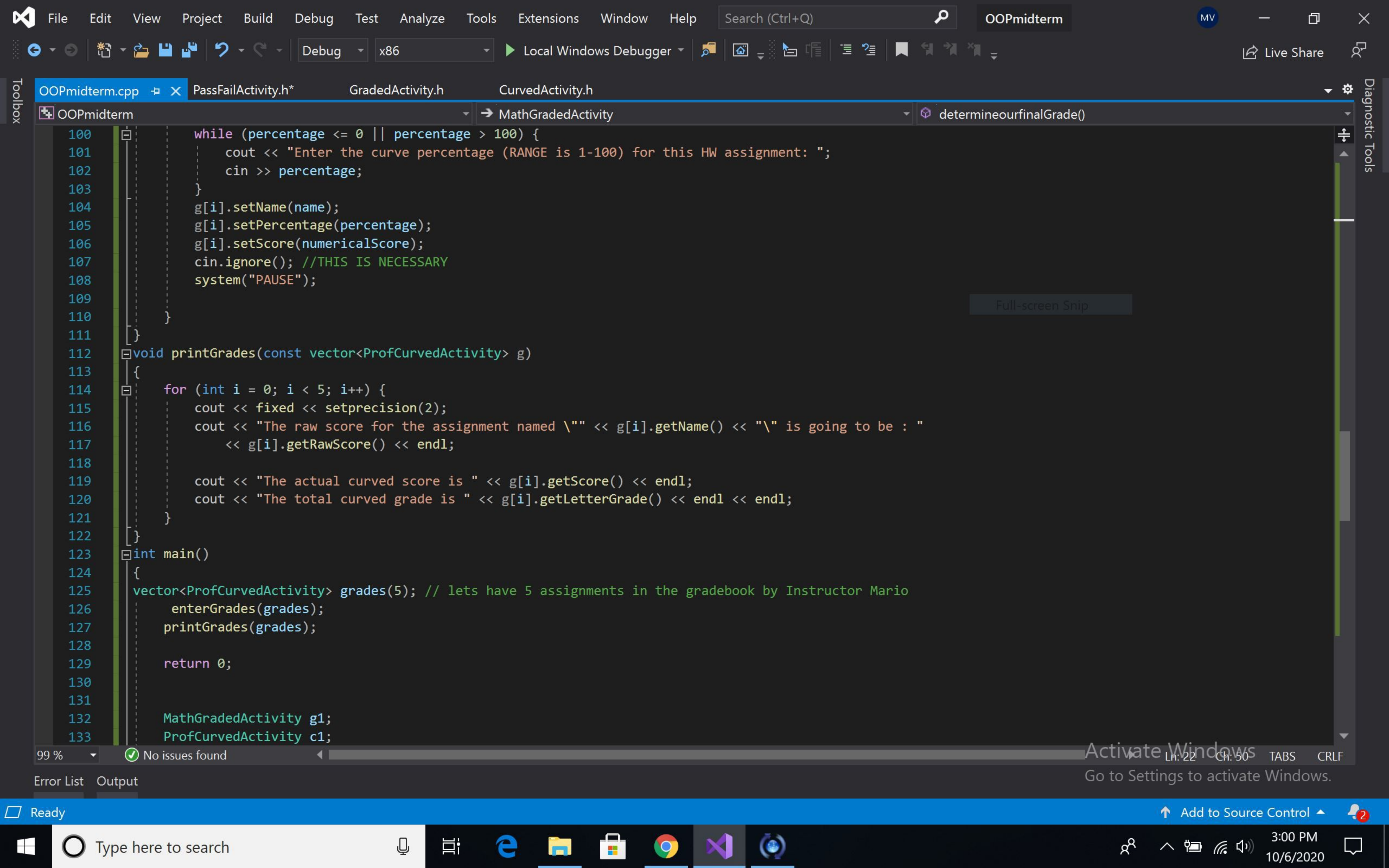
```
// OOPmidterm.cpp : This file contains the 'main' function. Program execution begins and ends there.
// Mario Valdez -- OOP final final assignment -- 10/5/20

/*Prompt:Modify the provided program we developed in the class, use main2()
and don't forget to specify virtual function for getLetterGrade() in GradedActivity.h and in PassFailActivity.h.*/

#include <iostream>
#include <vector>
#include <string>
#include <fstream>
#include <sstream>
#include <iomanip>
#include <cmath>
#include <algorithm>
#include "CurvedActivity.h"
#include "GradedActivity.h"
#include "PassFailActivity.h"
using namespace std;

void displayGradePlease(const MathGradedActivity&);  //new line

void MathGradedActivity::determineourfinalGrade()
{
    if (regularscore >= 90) {
        lettergrade = 'A';
        return;
    }
    if (regularscore >= 80) {
        lettergrade = 'B';
        return;
    }
    if (regularscore >= 70) {
        lettergrade = 'C';
        return;
```



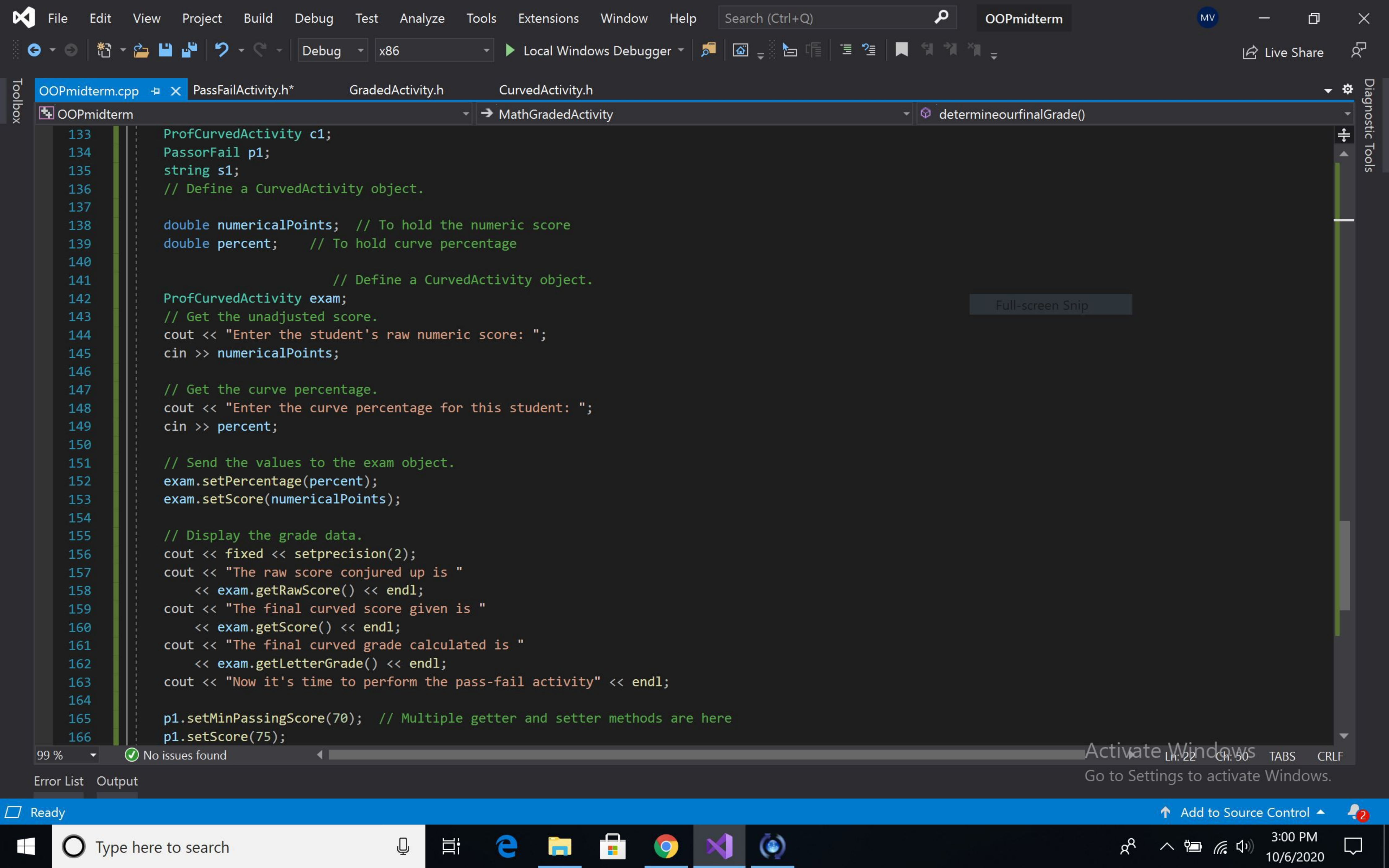
```cpp
        return;
    }
    if (regularscore >= 60) {   // new line
        lettergrade = 'D';  // neww line
            return; // new line
    }
    if (regularscore >= 50) {   /// new line
        lettergrade = 'E';  //new line
        return; //new line
    }
    if (regularscore >= 10) {  //new line
        lettergrade = 'Z';  //new line
        return;  //new line
    }

    lettergrade = 'F';
    return; // If we can avoid F's, then that is a lot less paperwork on our hands(but for the modified program, F's will do just fine)
}
void MathGradedActivity::setScore(double s)
{
    if (s > 100)
        s = 100; // no scores greater than 100
    regularscore = s;

    determineourfinalGrade();
}
void ProfCurvedActivity::setScore(double s)
{
    if (s > 100)
        s = 100; // we should keep to 100 - 0 range in order to simplify our solution

    rawScore = s;
    MathGradedActivity::setScore(rawScore * (1 + (rawPercentage / 100))); // this is the additive property using multiplication
}
```



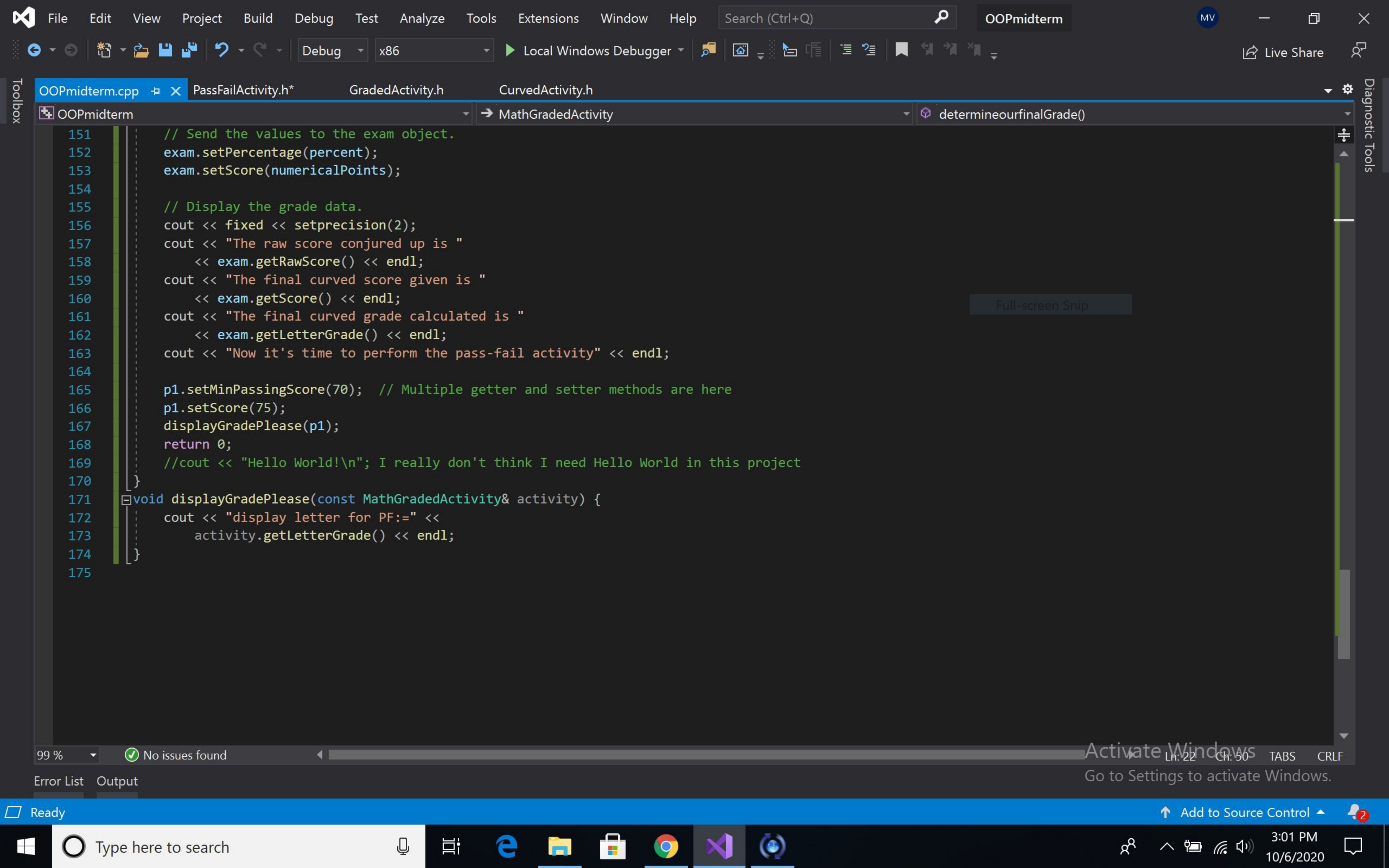
```cpp
}
void ProfCurvedActivity::setPercentage(double p)
{

    rawPercentage = p;
}
void ProfCurvedActivity::setName(string n)
{
    studentname = n;
}
string ProfCurvedActivity::getName() const
{
    return studentname;
}
void enterGrades(vector<ProfCurvedActivity>& g)
{
    float numericalScore; // To hold the numeric score of the assignment
    float percentage; // To hold curve percentage as held preference by the instructor
    string name;
    // Get the unadjusted score.
    for (int i = 0; i < 5; i++) {
        cout << "Enter the name for this homework assignment due today: ";
        getline(cin, name); //this was getting skipped on second iteration
        //maybe needs a chuck on the length?

        cout << "Enter the student's raw numeric score for this homework assignment so that we may begin calculations: ";
        cin >> numericalScore; // this is the output for the numerical score


        // Get the curve percentage.
        cout << "Enter the curve percentage (RANGE is 1-100) for this hw assignment: ";
        cin >> percentage;

        while (percentage <= 0 || percentage > 100) {
```



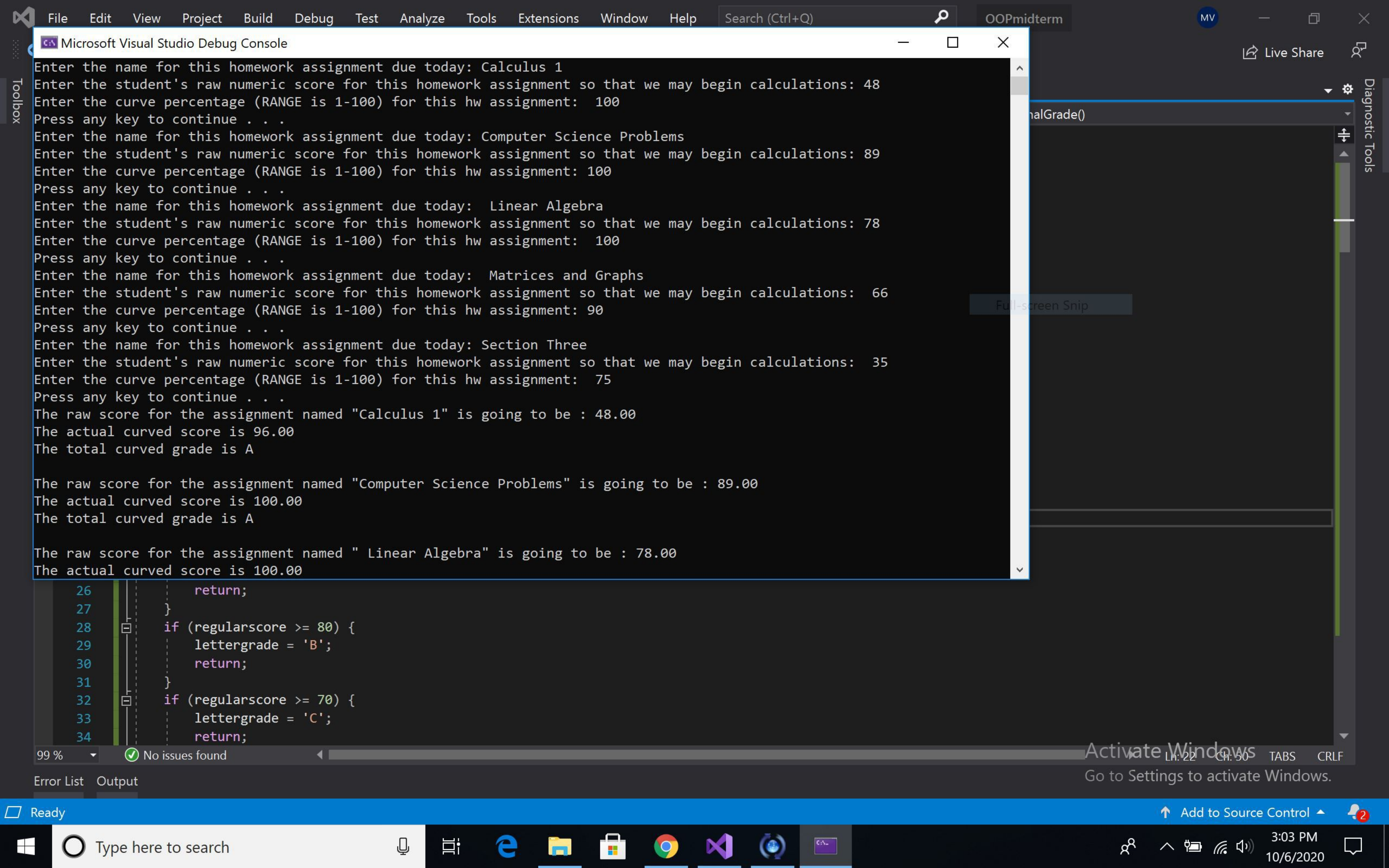
```
        while (percentage <= 0 || percentage > 100) {
            cout << "Enter the curve percentage (RANGE is 1-100) for this HW assignment: ";
            cin >> percentage;
        }
        g[i].setName(name);
        g[i].setPercentage(percentage);
        g[i].setScore(numericalScore);
        cin.ignore(); //THIS IS NECESSARY
        system("PAUSE");

    }
}
void printGrades(const vector<ProfCurvedActivity> g)
{
    for (int i = 0; i < 5; i++) {
        cout << fixed << setprecision(2);
        cout << "The raw score for the assignment named \"" << g[i].getName() << "\" is going to be : "
            << g[i].getRawScore() << endl;

        cout << "The actual curved score is " << g[i].getScore() << endl;
        cout << "The total curved grade is " << g[i].getLetterGrade() << endl << endl;
    }
}
int main()
{
vector<ProfCurvedActivity> grades(5); // lets have 5 assignments in the gradebook by Instructor Mario
     enterGrades(grades);
    printGrades(grades);

    return 0;


    MathGradedActivity g1;
    ProfCurvedActivity c1;
```



```
    ProfCurvedActivity c1;
    PassorFail p1;
    string s1;
    // Define a CurvedActivity object.

    double numericalPoints;  // To hold the numeric score
    double percent;    // To hold curve percentage

                          // Define a CurvedActivity object.
    ProfCurvedActivity exam;
    // Get the unadjusted score.
    cout << "Enter the student's raw numeric score: ";
    cin >> numericalPoints;

    // Get the curve percentage.
    cout << "Enter the curve percentage for this student: ";
    cin >> percent;

    // Send the values to the exam object.
    exam.setPercentage(percent);
    exam.setScore(numericalPoints);

    // Display the grade data.
    cout << fixed << setprecision(2);
    cout << "The raw score conjured up is "
        << exam.getRawScore() << endl;
    cout << "The final curved score given is "
        << exam.getScore() << endl;
    cout << "The final curved grade calculated is "
        << exam.getLetterGrade() << endl;
    cout << "Now it's time to perform the pass-fail activity" << endl;

    p1.setMinPassingScore(70);  // Multiple getter and setter methods are here
    p1.setScore(75);
```

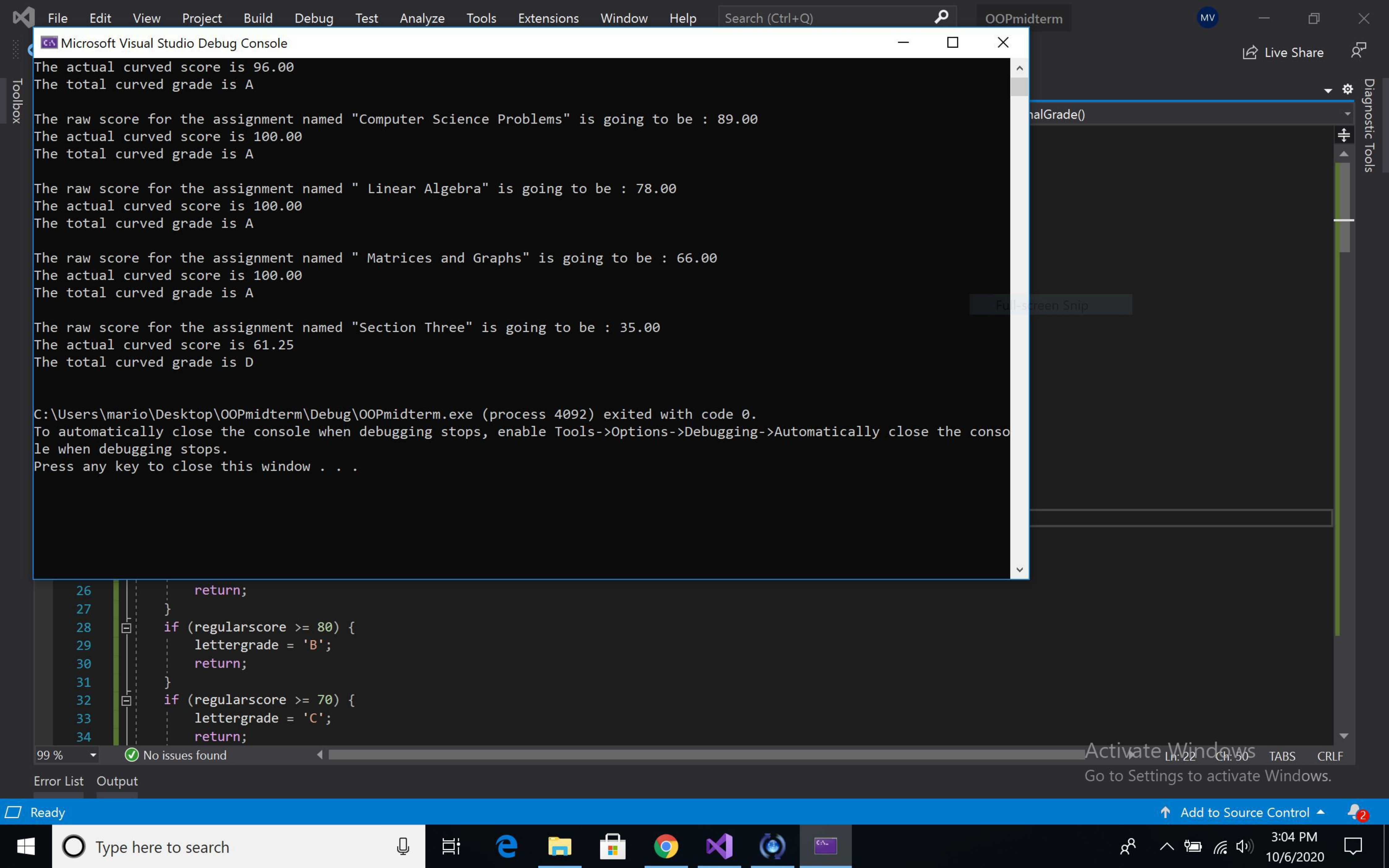


```cpp
    // Send the values to the exam object.
    exam.setPercentage(percent);
    exam.setScore(numericalPoints);

    // Display the grade data.
    cout << fixed << setprecision(2);
    cout << "The raw score conjured up is "
        << exam.getRawScore() << endl;
    cout << "The final curved score given is "
        << exam.getScore() << endl;
    cout << "The final curved grade calculated is "
        << exam.getLetterGrade() << endl;
    cout << "Now it's time to perform the pass-fail activity" << endl;

    p1.setMinPassingScore(70);  // Multiple getter and setter methods are here
    p1.setScore(75);
    displayGradePlease(p1);
    return 0;
    //cout << "Hello World!\n"; I really don't think I need Hello World in this project
}
void displayGradePlease(const MathGradedActivity& activity) {
    cout << "display letter for PF:=" <<
        activity.getLetterGrade() << endl;
}

```



```
Enter the name for this homework assignment due today: Calculus 1
Enter the student's raw numeric score for this homework assignment so that we may begin calculations: 48
Enter the curve percentage (RANGE is 1-100) for this hw assignment:  100
Press any key to continue . . .
Enter the name for this homework assignment due today: Computer Science Problems
Enter the student's raw numeric score for this homework assignment so that we may begin calculations: 89
Enter the curve percentage (RANGE is 1-100) for this hw assignment: 100
Press any key to continue . . .
Enter the name for this homework assignment due today:  Linear Algebra
Enter the student's raw numeric score for this homework assignment so that we may begin calculations: 78
Enter the curve percentage (RANGE is 1-100) for this hw assignment:  100
Press any key to continue . . .
Enter the name for this homework assignment due today:  Matrices and Graphs
Enter the student's raw numeric score for this homework assignment so that we may begin calculations:  66
Enter the curve percentage (RANGE is 1-100) for this hw assignment: 90
Press any key to continue . . .
Enter the name for this homework assignment due today: Section Three
Enter the student's raw numeric score for this homework assignment so that we may begin calculations:  35
Enter the curve percentage (RANGE is 1-100) for this hw assignment:  75
Press any key to continue . . .
The raw score for the assignment named "Calculus 1" is going to be : 48.00
The actual curved score is 96.00
The total curved grade is A

The raw score for the assignment named "Computer Science Problems" is going to be : 89.00
The actual curved score is 100.00
The total curved grade is A

The raw score for the assignment named " Linear Algebra" is going to be : 78.00
The actual curved score is 100.00
```

```
            return;
        }
        if (regularscore >= 80) {
            lettergrade = 'B';
            return;
        }
        if (regularscore >= 70) {
            lettergrade = 'C';
            return;
```



```
The actual curved score is 96.00
The total curved grade is A

The raw score for the assignment named "Computer Science Problems" is going to be : 89.00
The actual curved score is 100.00
The total curved grade is A

The raw score for the assignment named " Linear Algebra" is going to be : 78.00
The actual curved score is 100.00
The total curved grade is A

The raw score for the assignment named " Matrices and Graphs" is going to be : 66.00
The actual curved score is 100.00
The total curved grade is A

The raw score for the assignment named "Section Three" is going to be : 35.00
The actual curved score is 61.25
The total curved grade is D


C:\Users\mario\Desktop\OOPmidterm\Debug\OOPmidterm.exe (process 4092) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

```
            return;
        }
        if (regularscore >= 80) {
            lettergrade = 'B';
            return;
        }
        if (regularscore >= 70) {
            lettergrade = 'C';
            return;
```