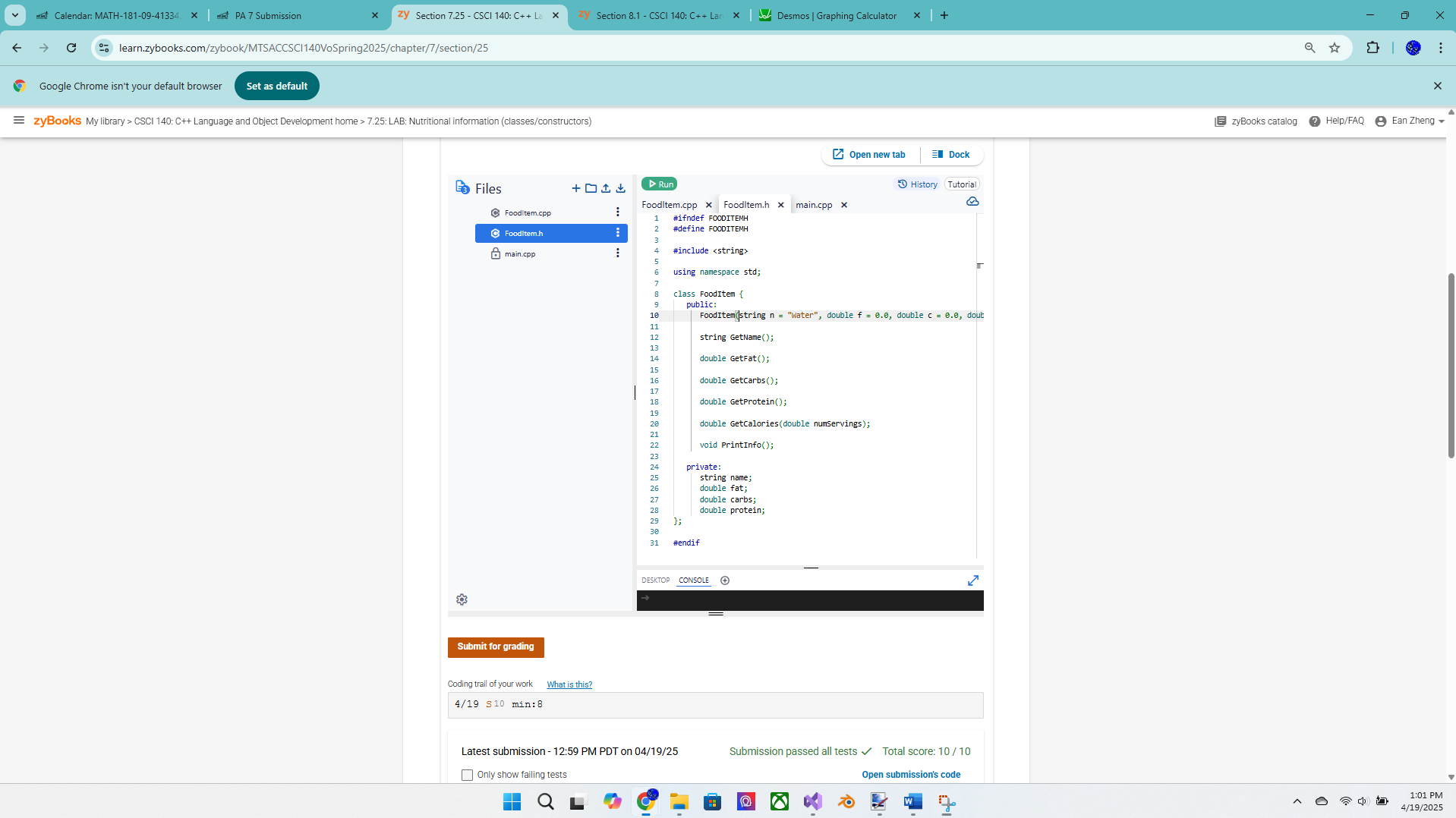
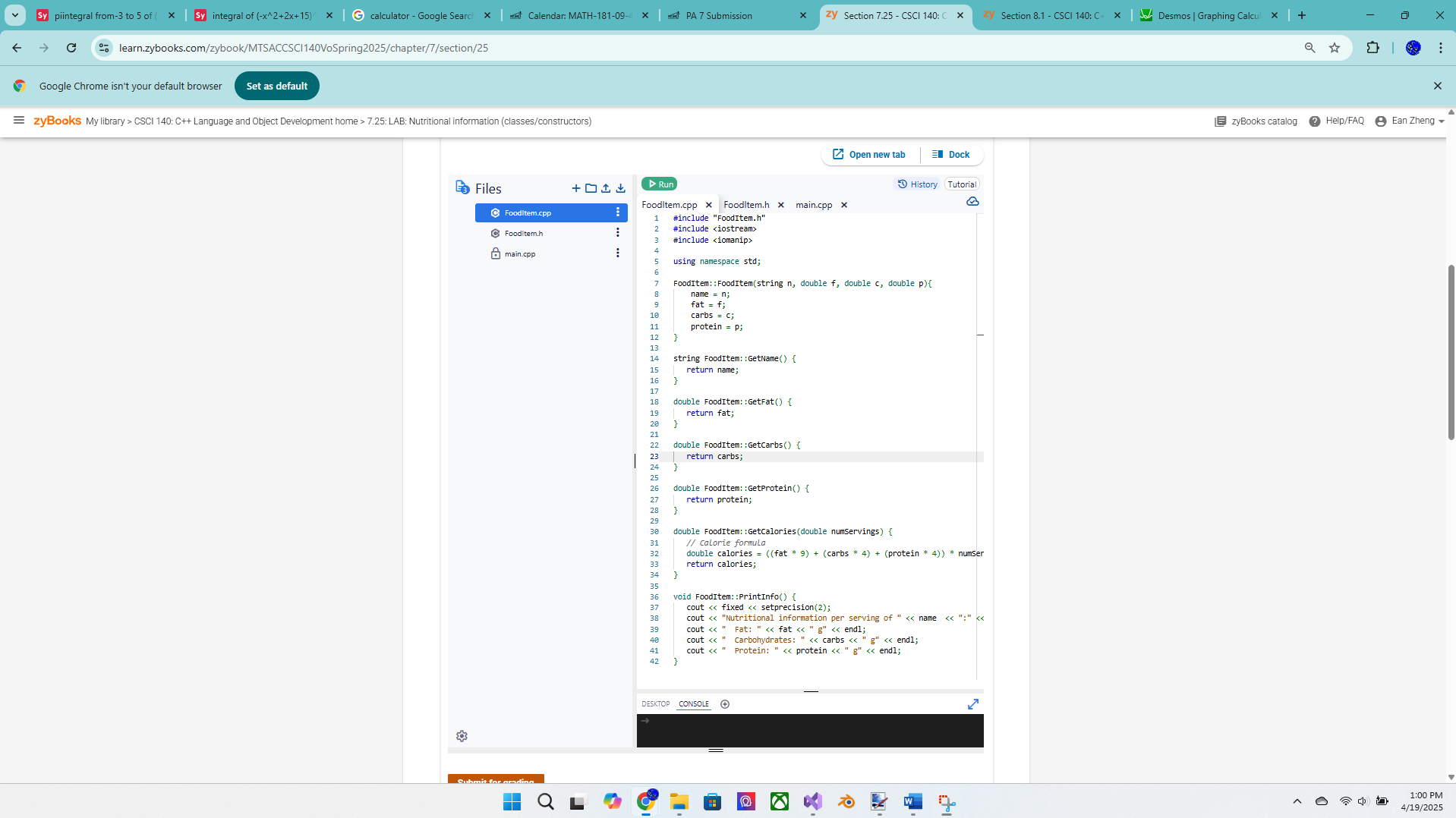
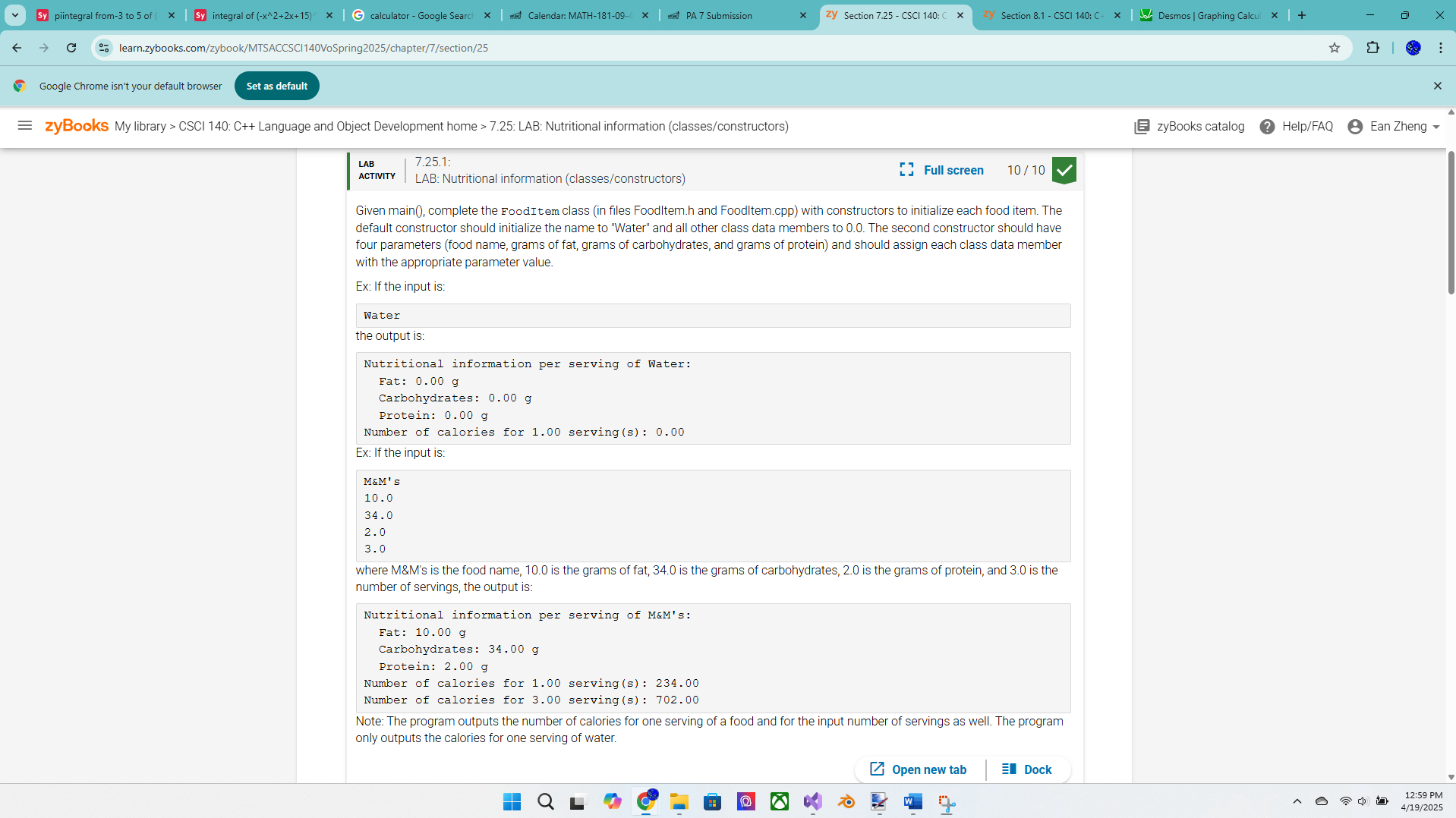
## CSCI 140 PA 7 Submission

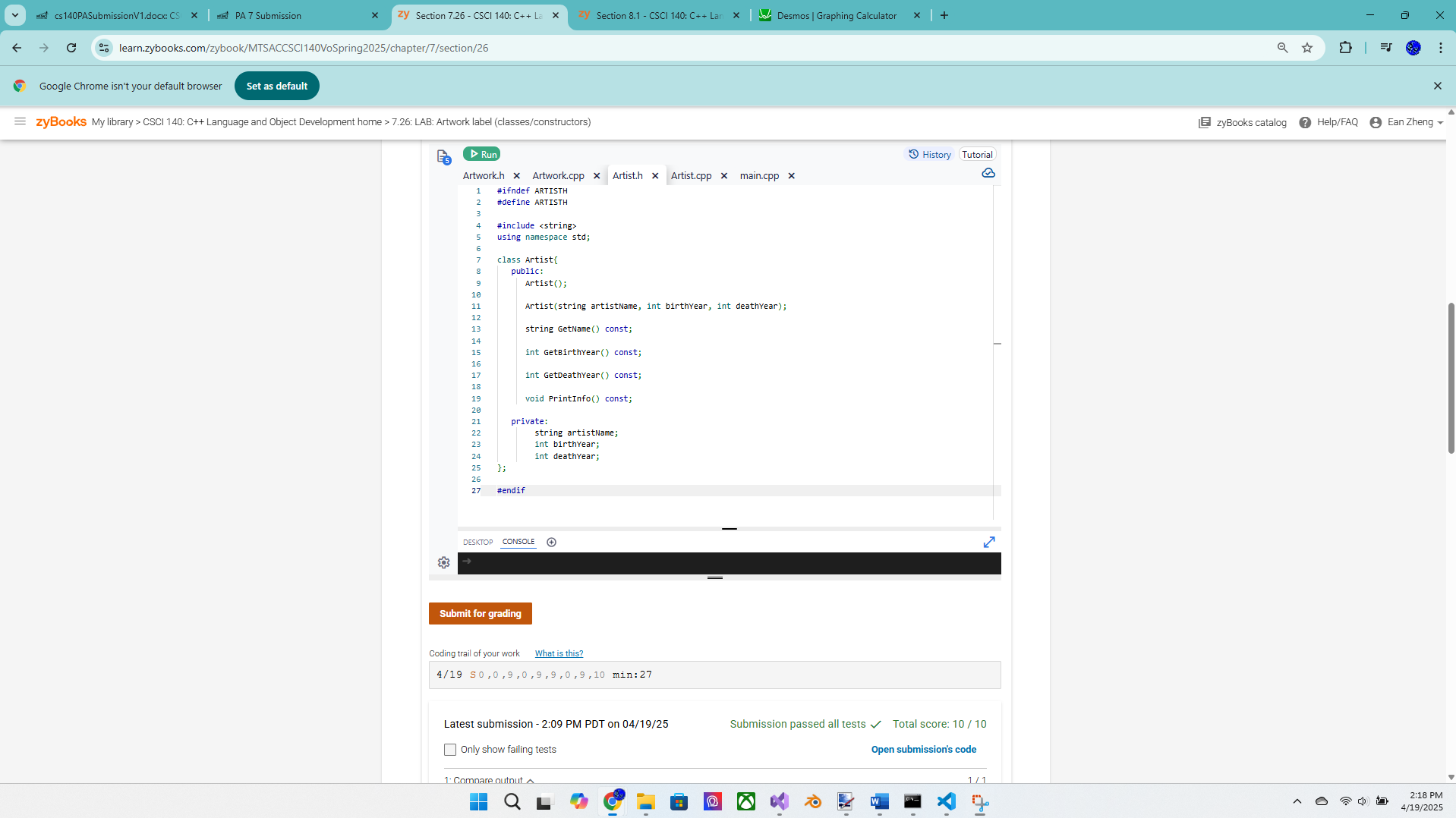
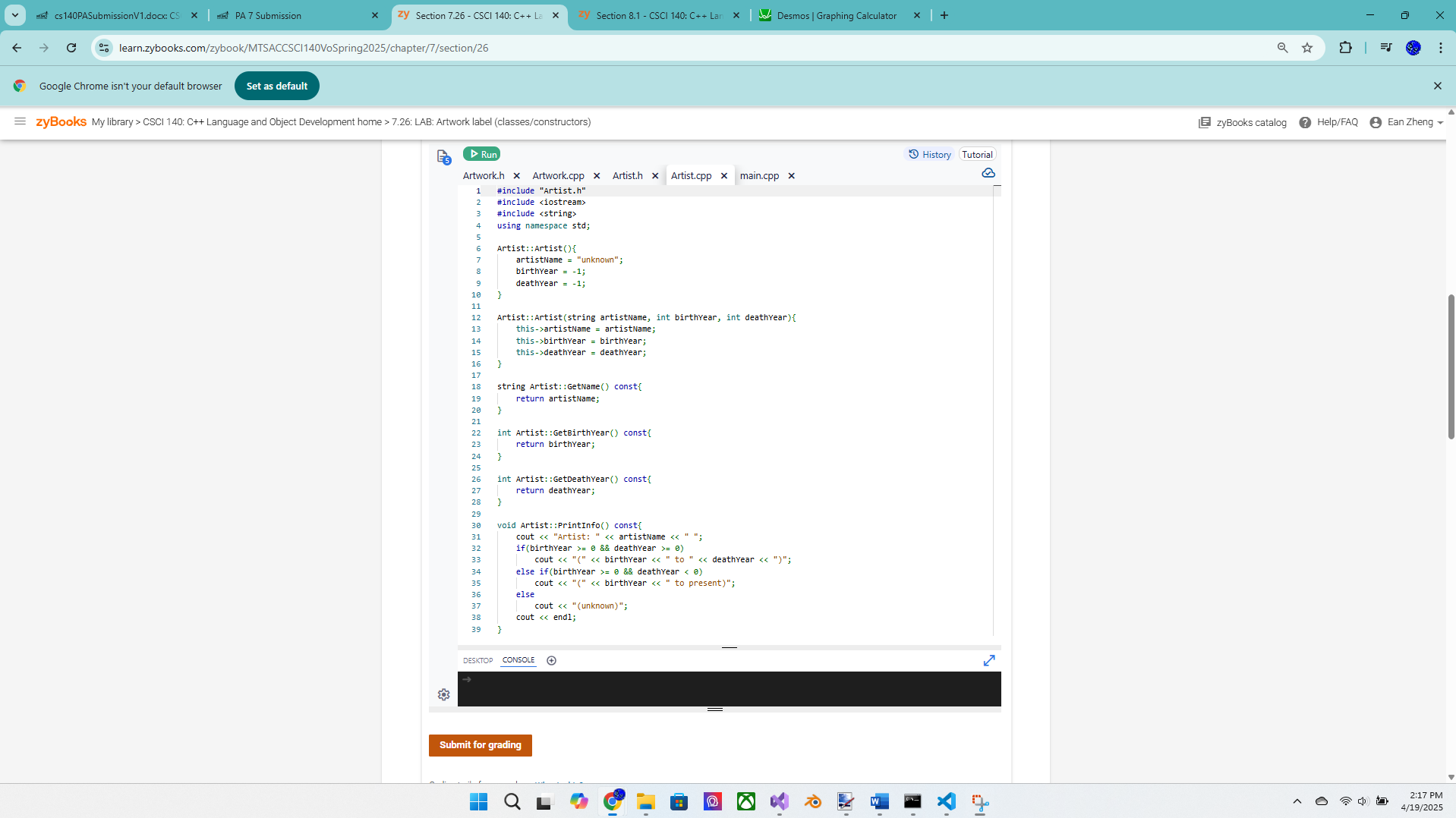
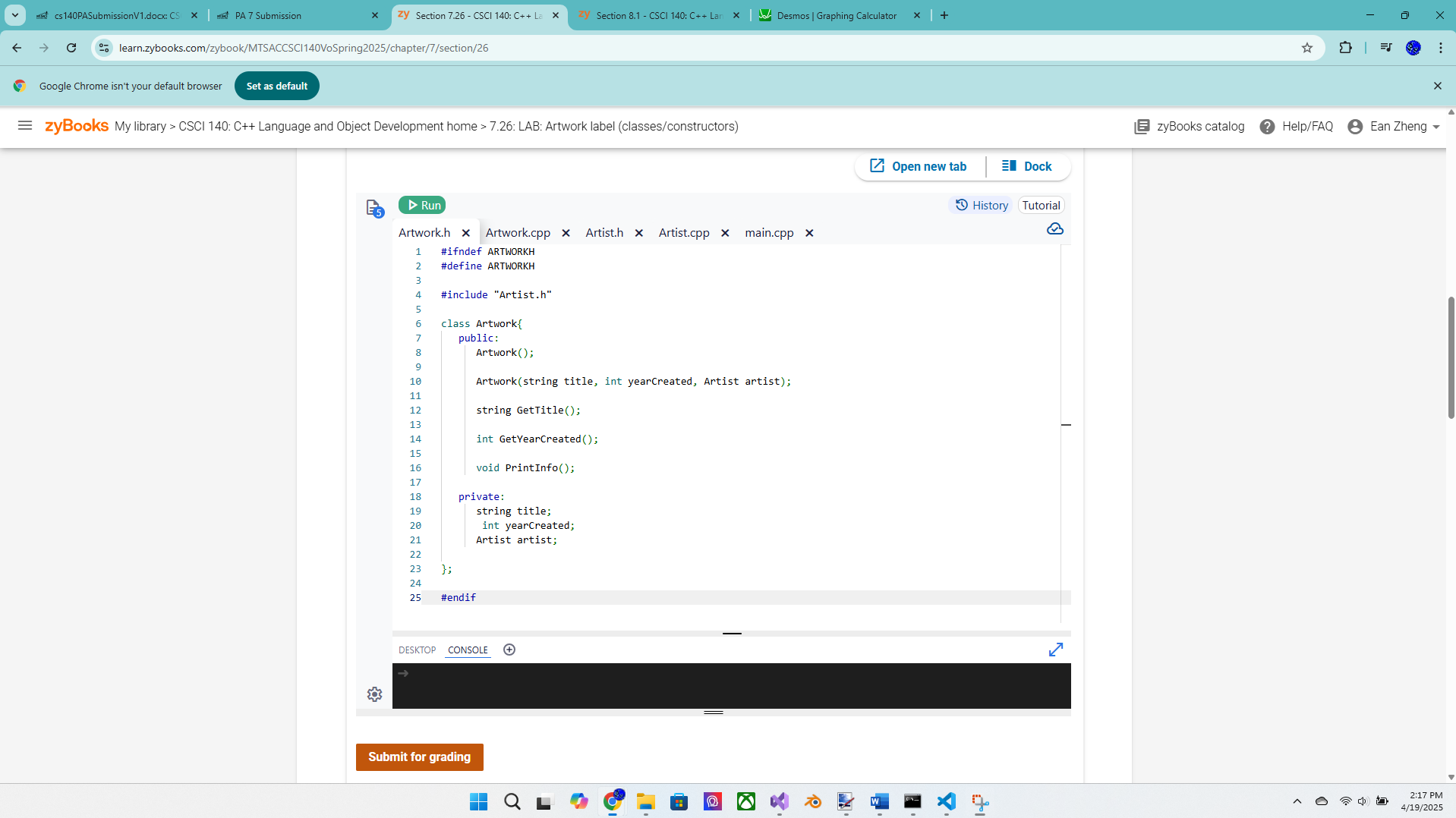
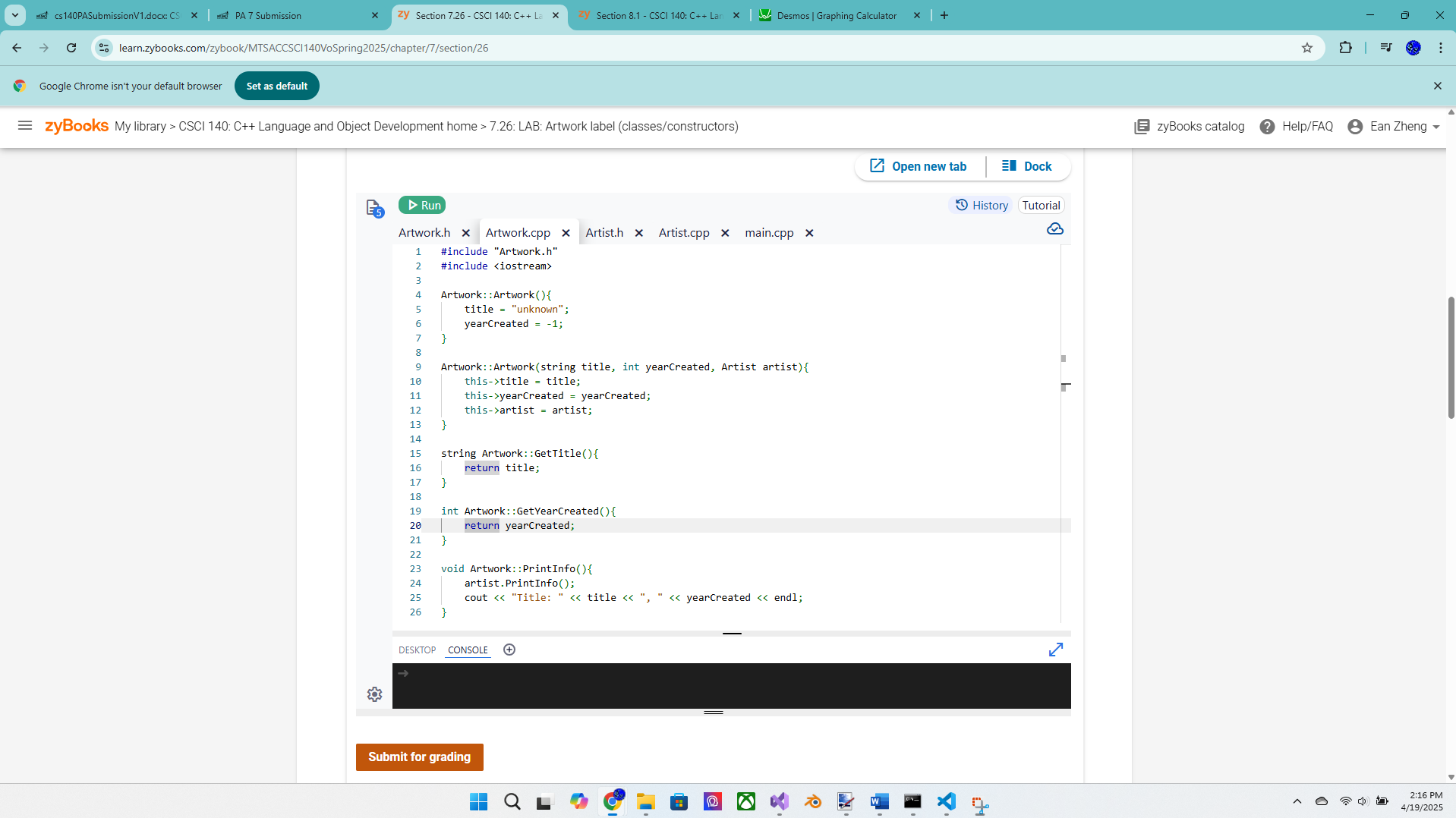
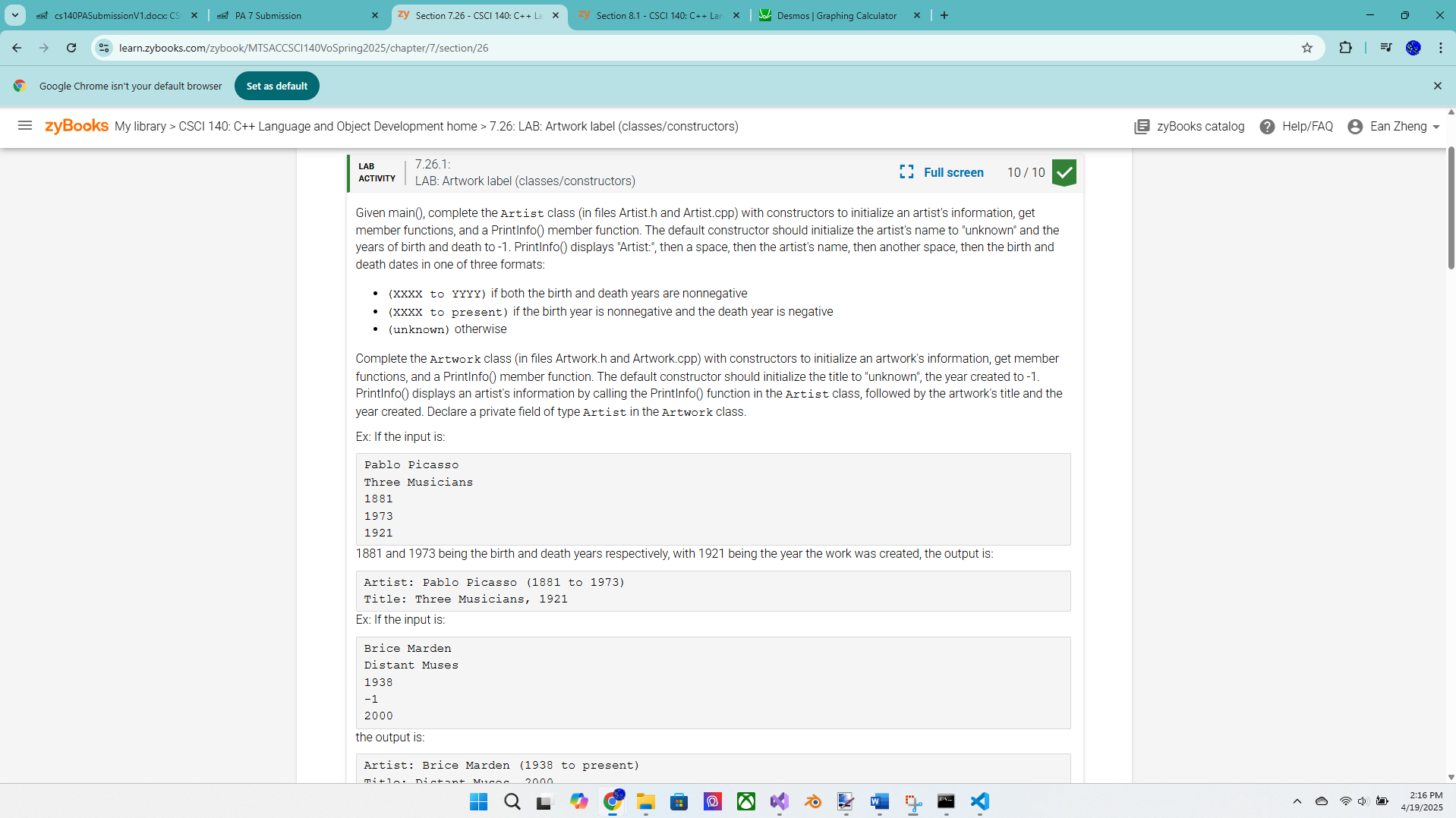
## Due Date: 4/21/2025

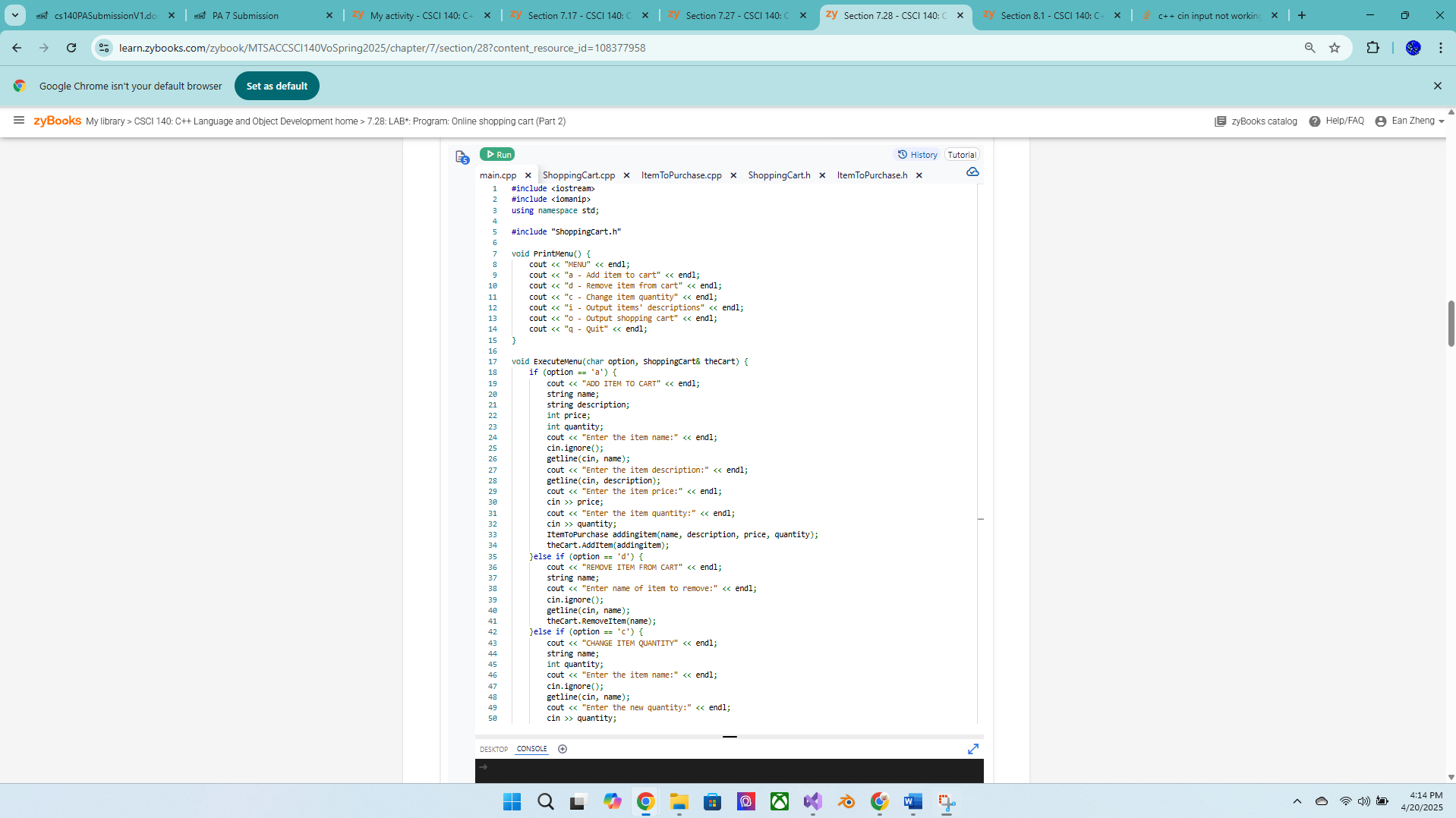
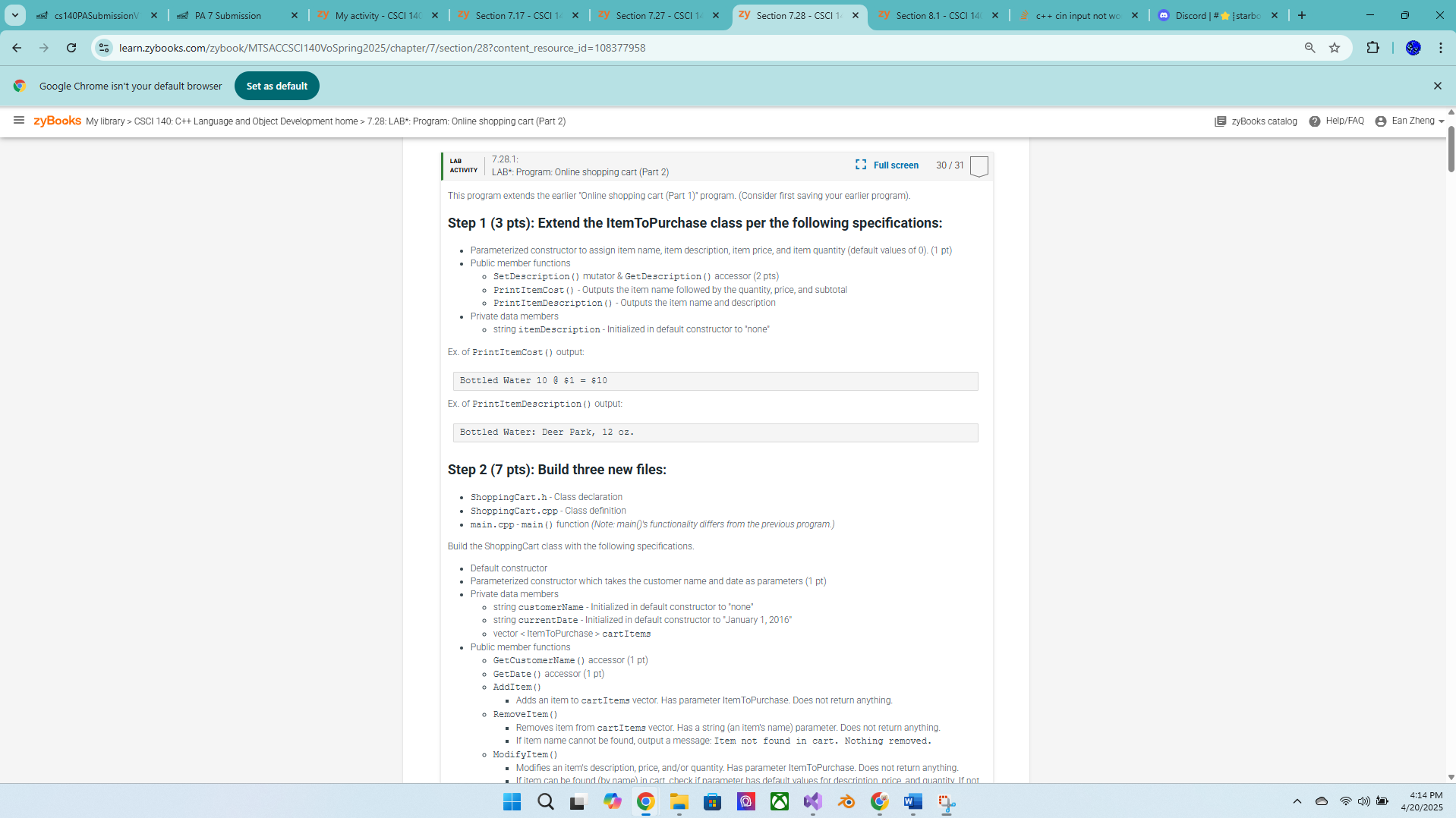
## Name(s): Ean Zheng

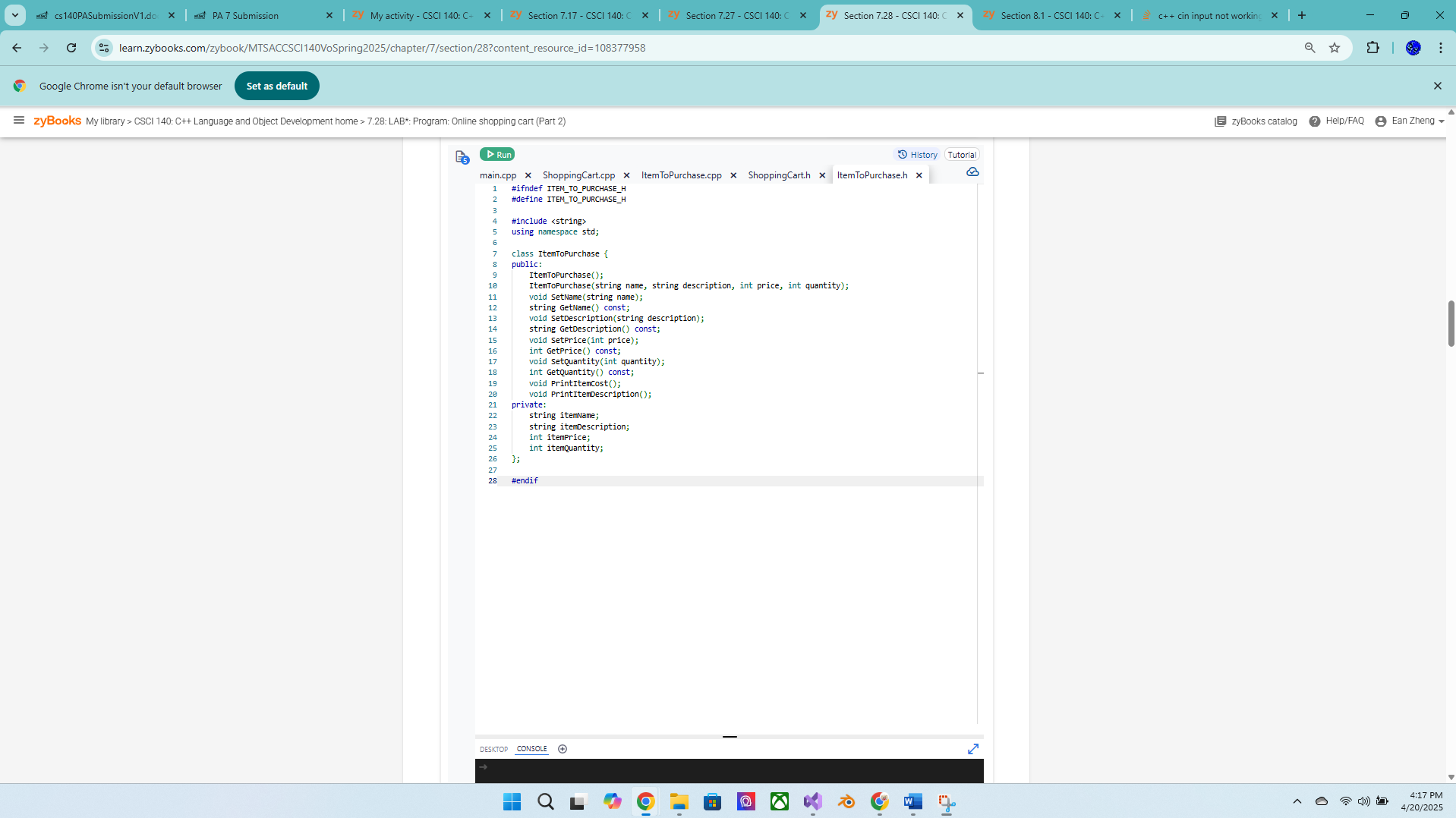
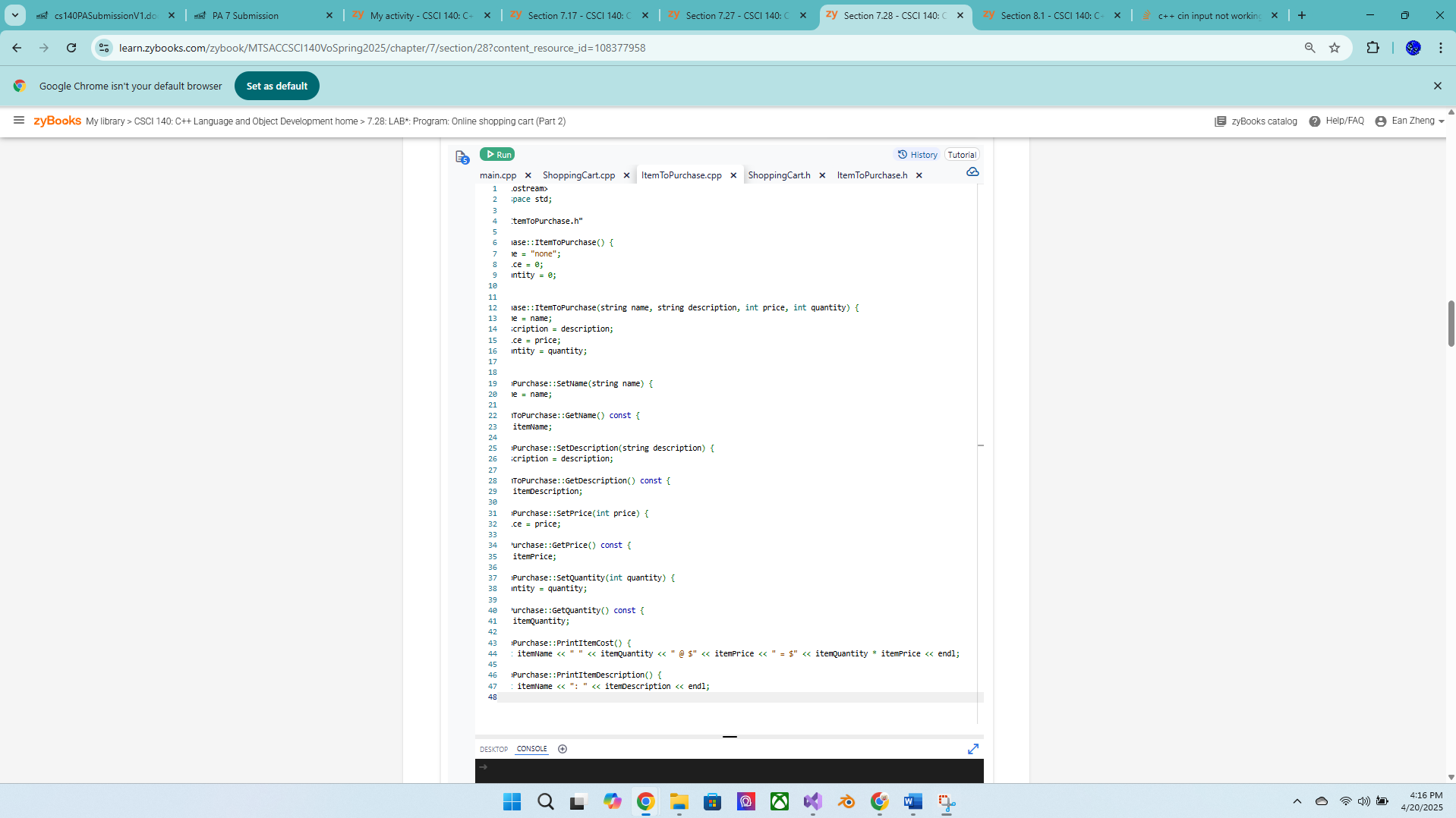
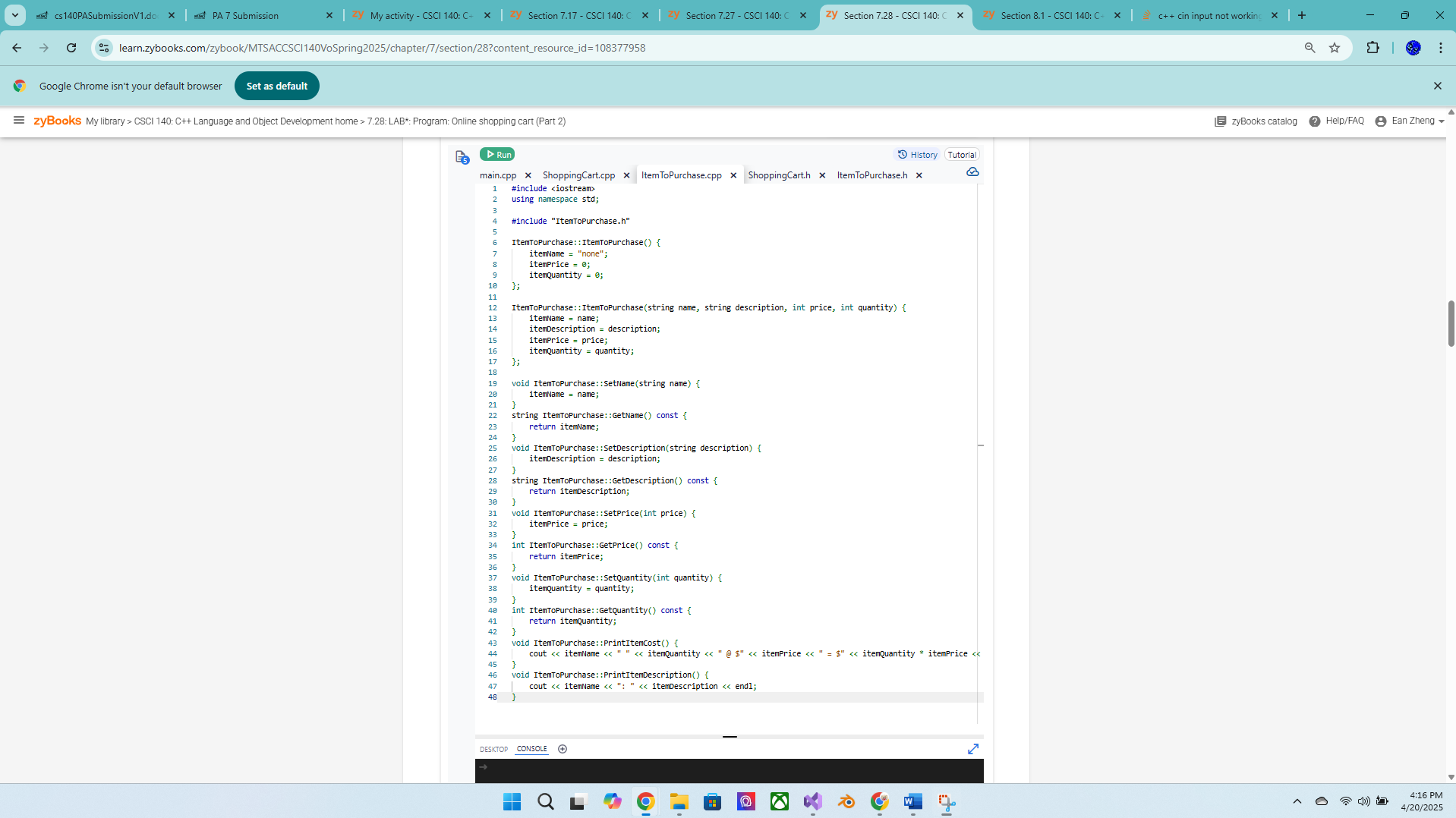
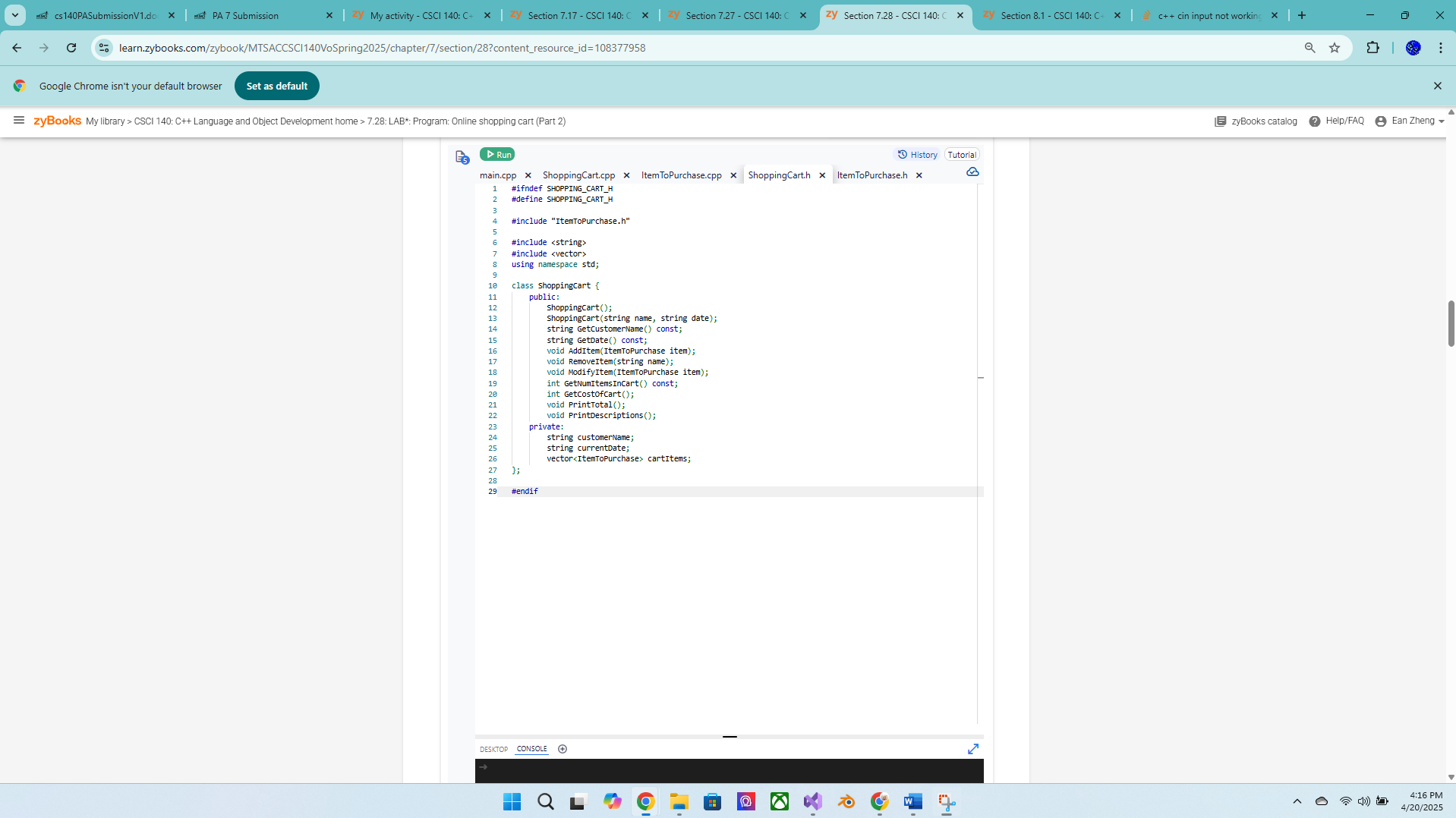
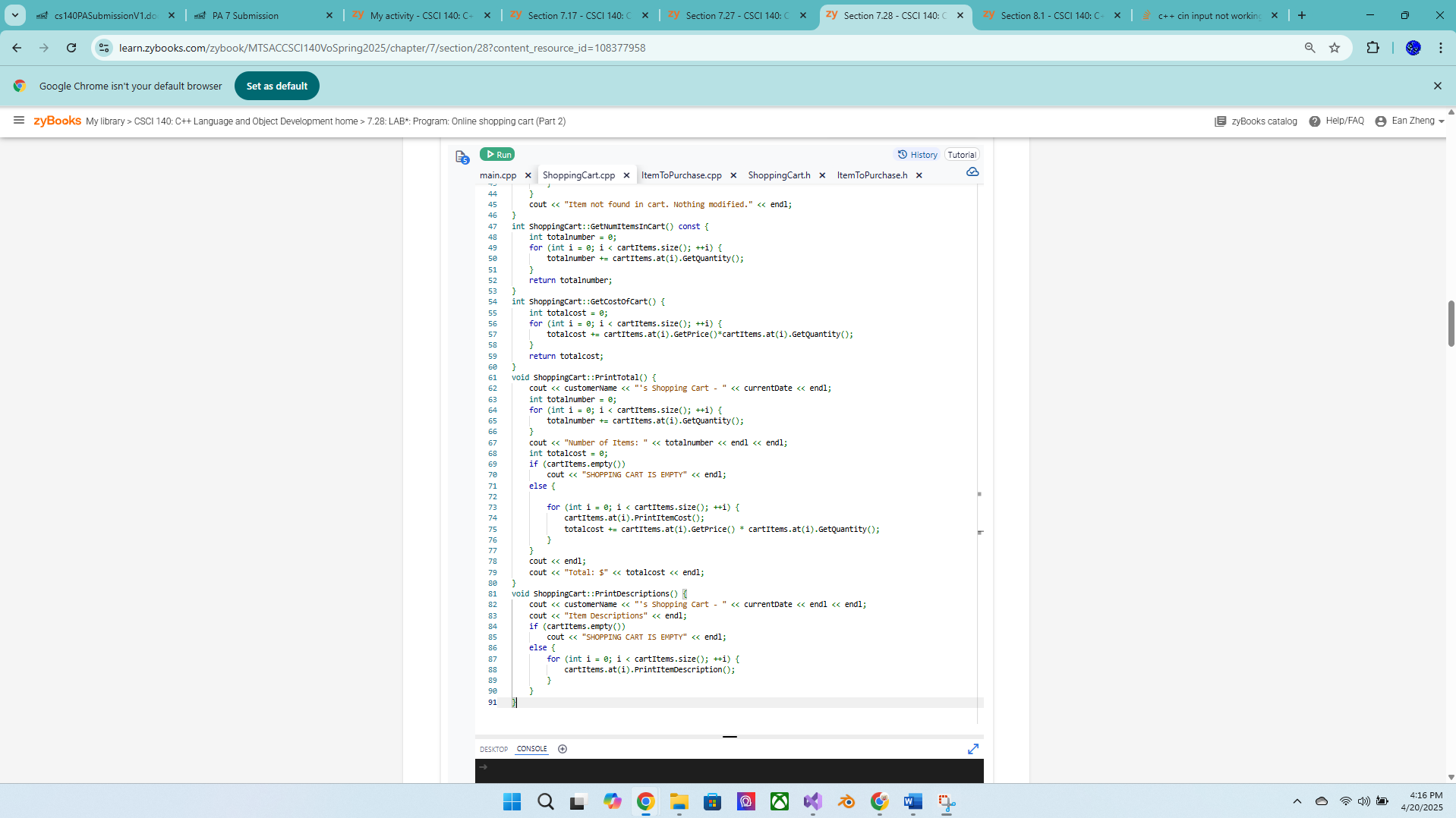
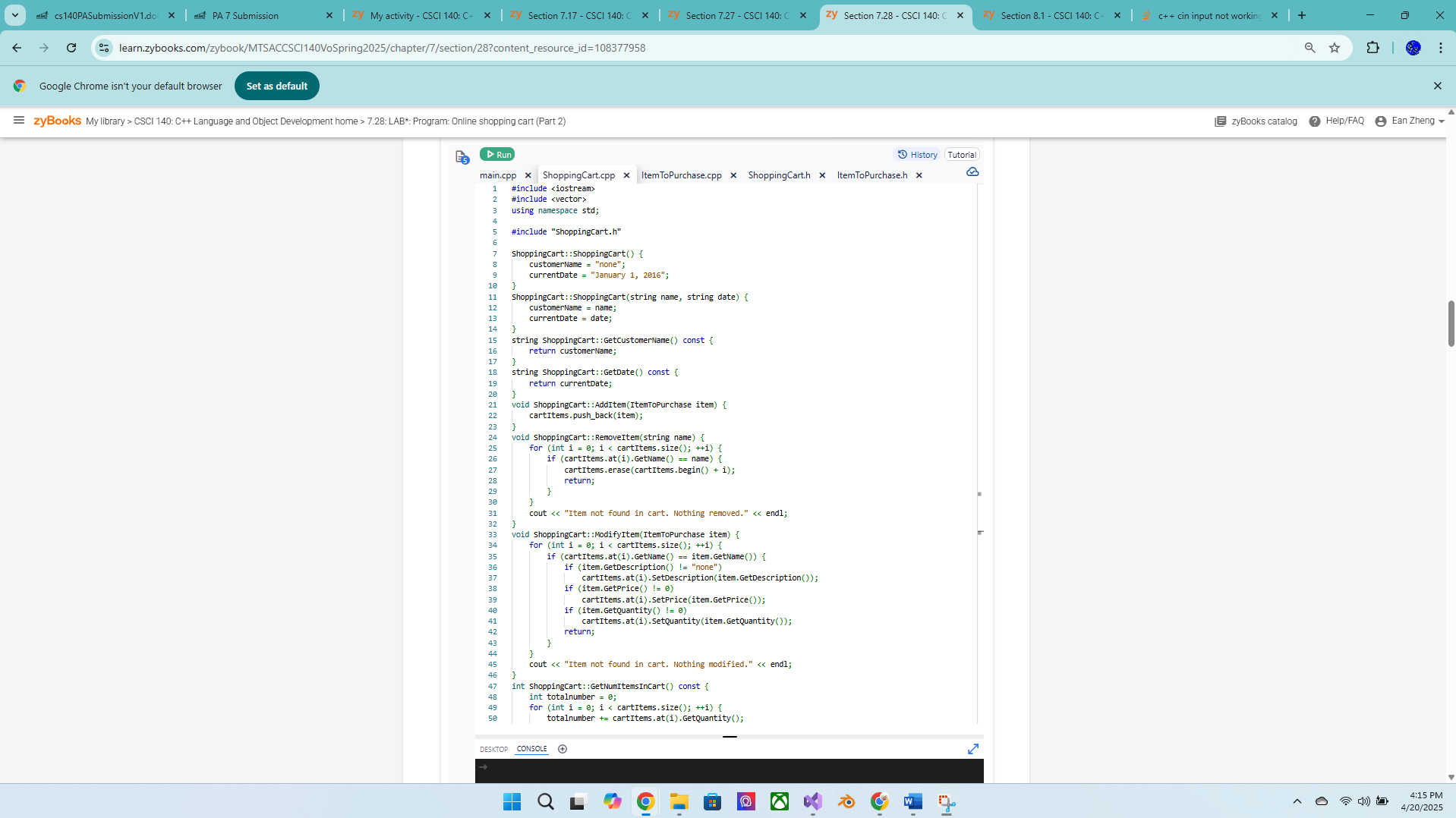
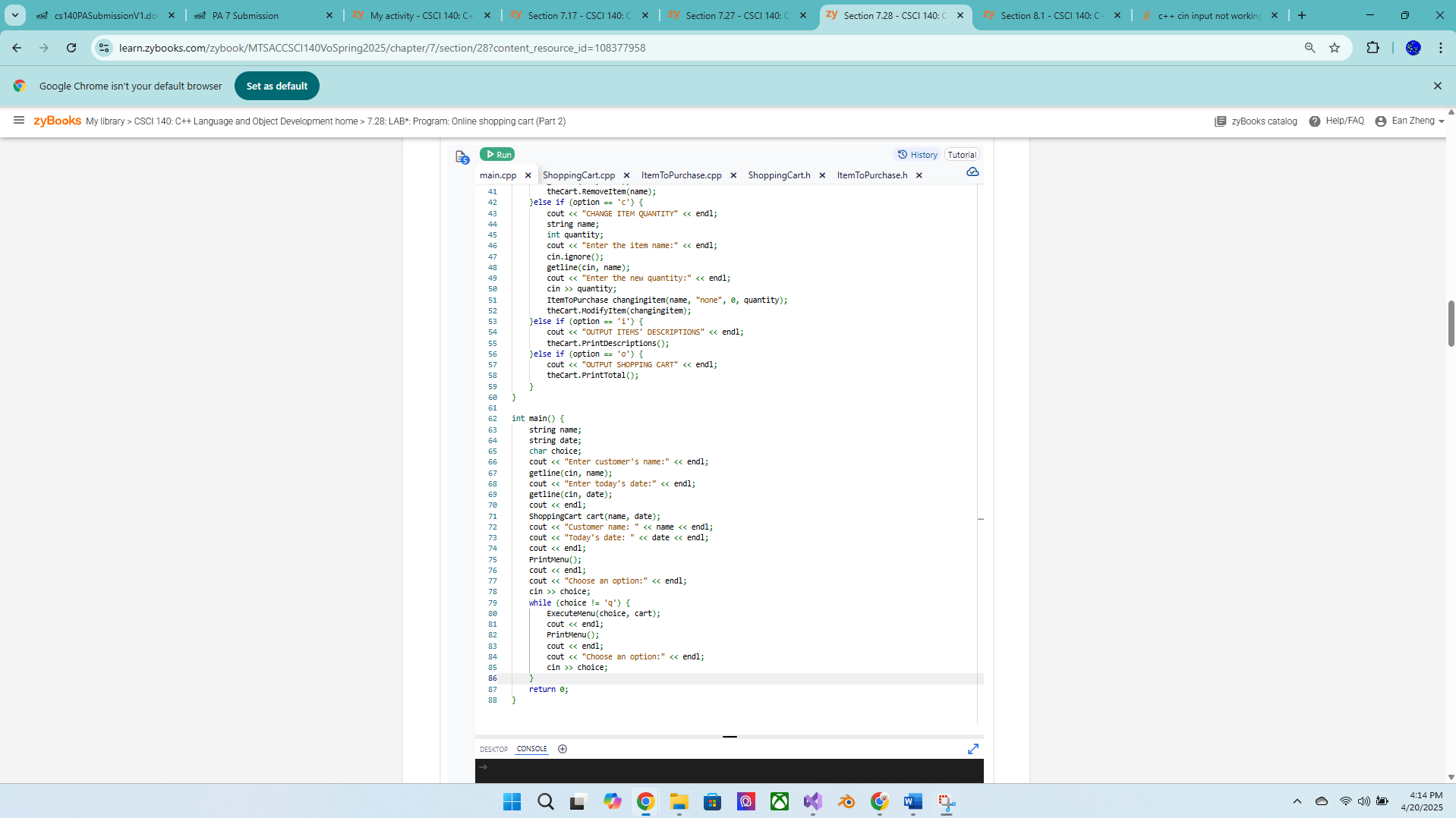
Exercise 1 – 7.25 LAB: Nutritional information (classes/constructors)



Exercise 2 – 7.26 LAB: Artwork label

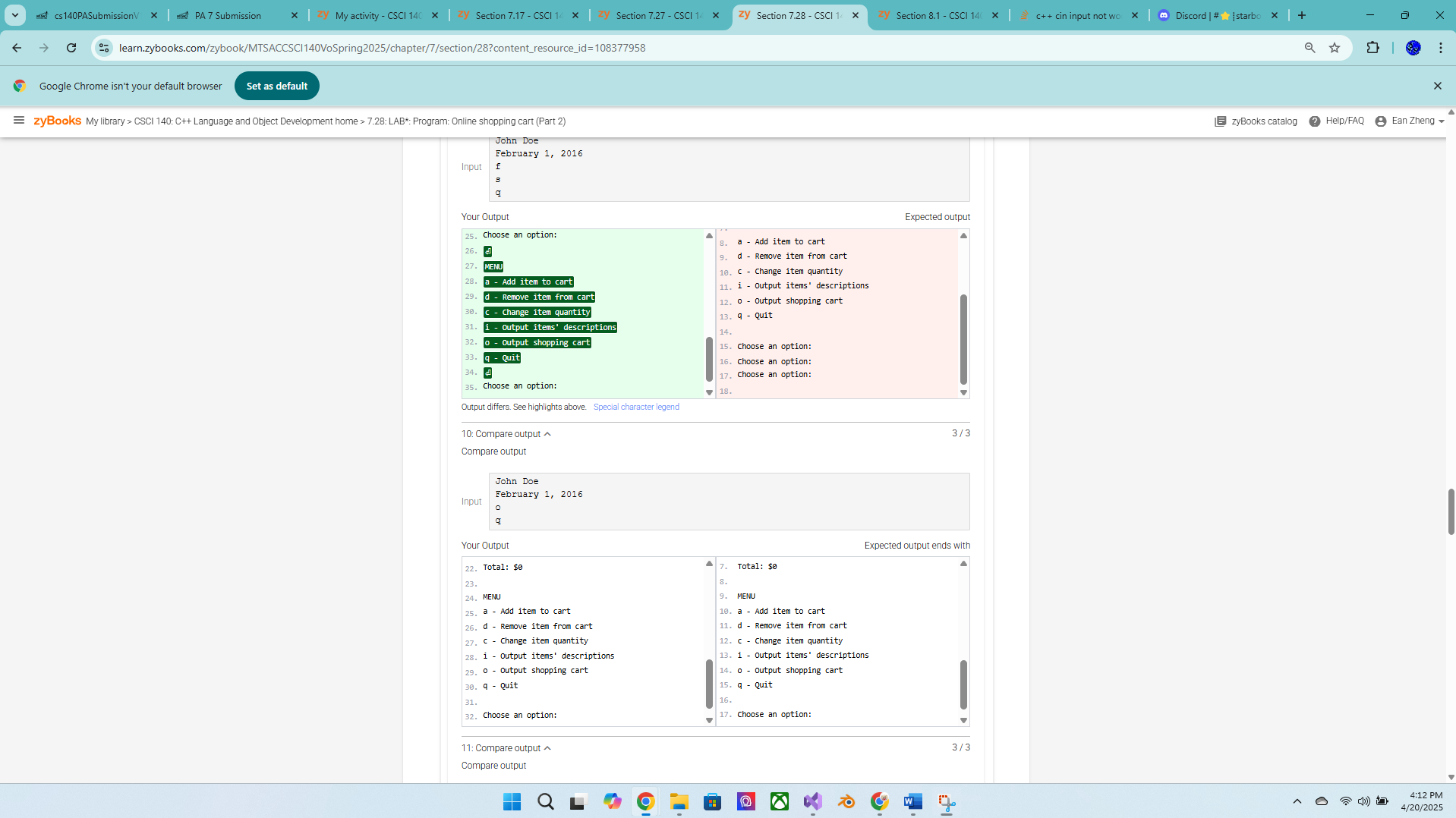


Exercise 3 – 7.28 LAB\*: Warm up: Online shopping cart (Part 2) – more points for this  
exercise  


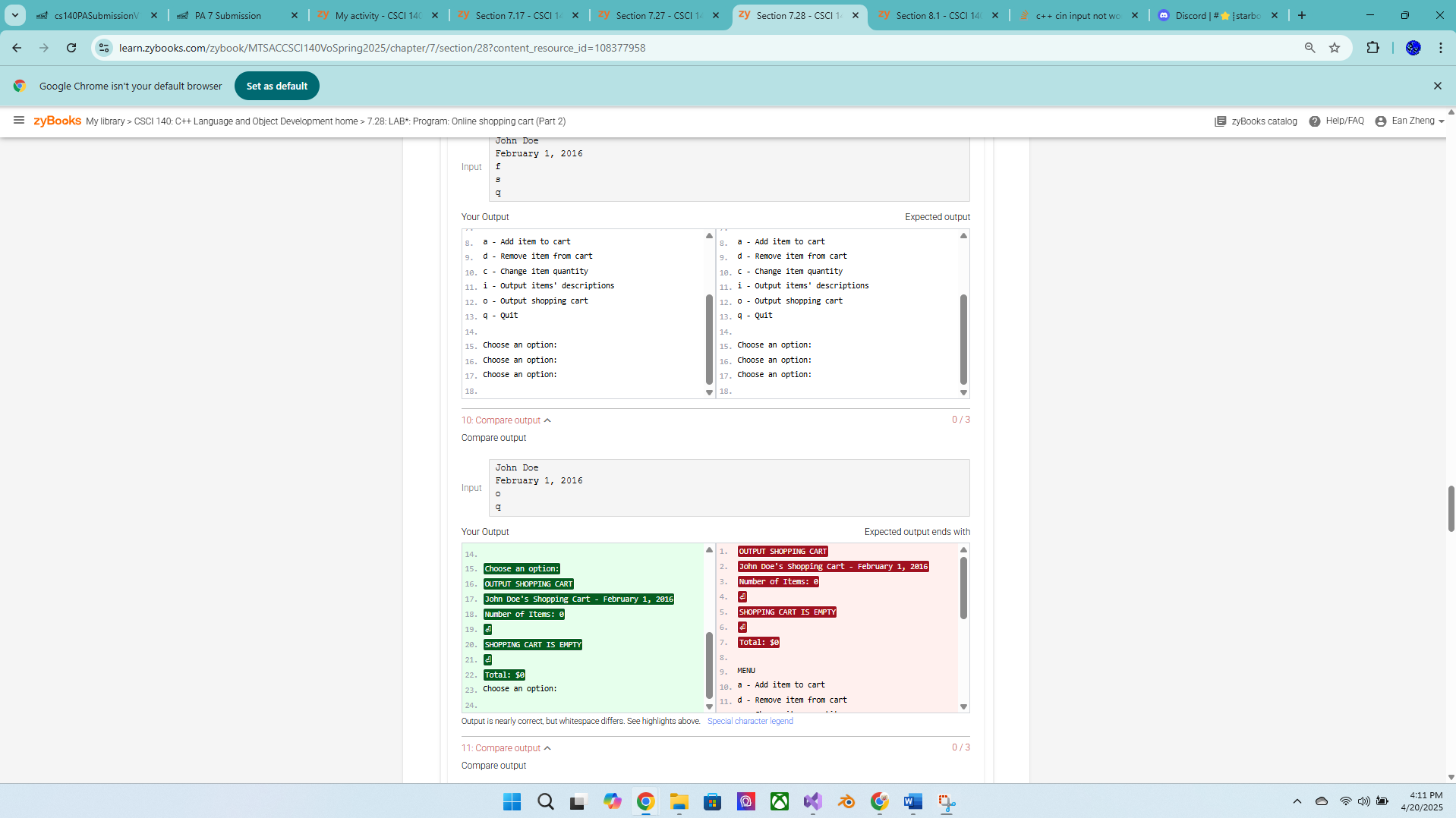
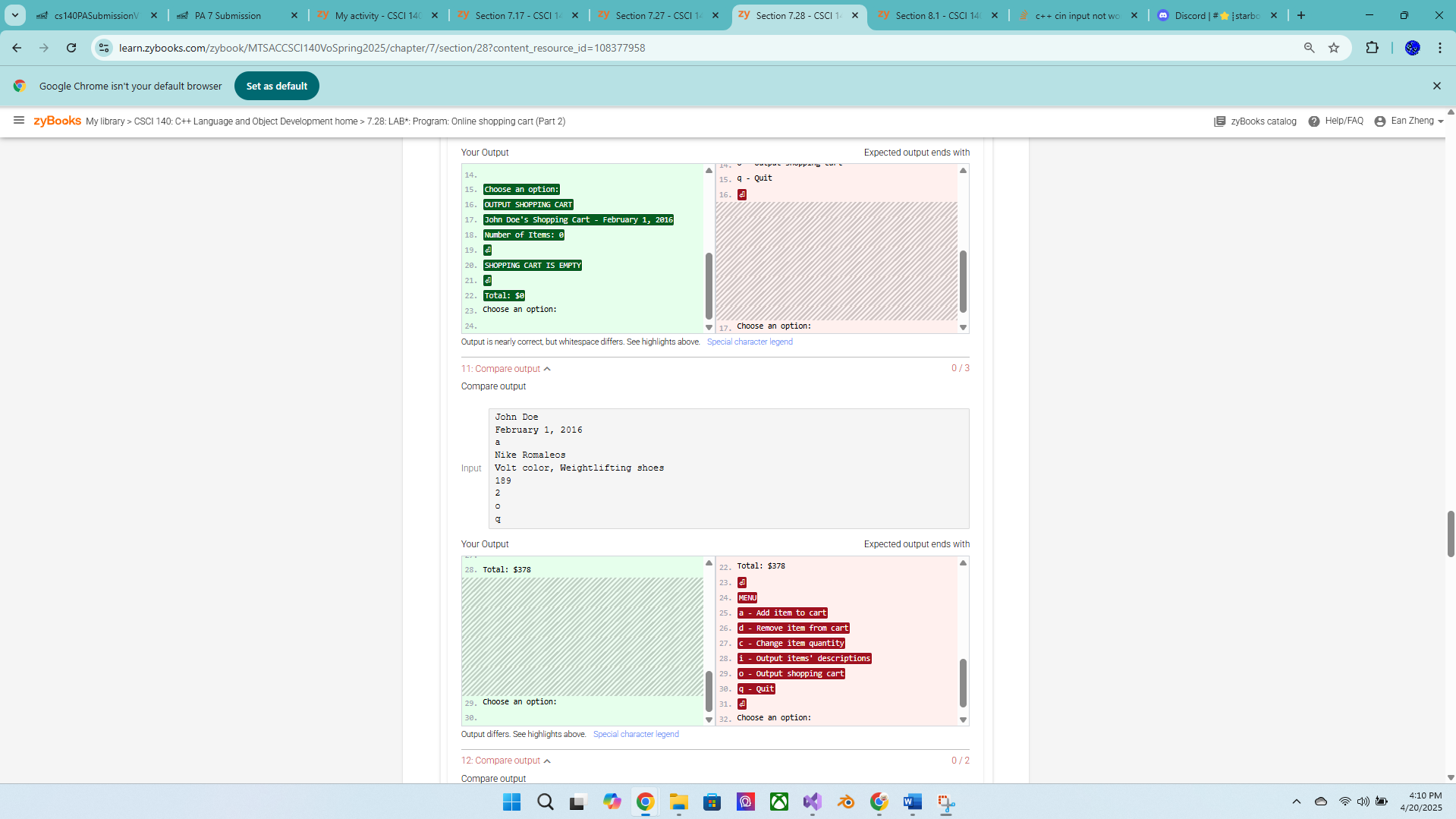


1 input case can’t be satisfied with the rest at the same time, the rest require printing menu every choice loop iteration while this one doesn’t; is website issue

1 Wrong because 1 test case isn’t satisfied from menu printing each choice loop iteration:



Many wrong because each choice loop iteration doesn’t print menu:



Exercise 4 – Height class version 2 – more points for this exercise  
Add the following operations to the Height class from previous lab and include  
additional test cases to add these three new features:  
• Overload advance function so now there are two advance functions  
o void increment(int inches);  
o // make sure inches must be between 1 and 11  
o // ignore invalid inches  
• Returning the total inches.  
o int totalInches() const;  
• Overload operator == to compare the two heights. Can use either a member  
function or a friend function  
o bool operator==(const Height &r) const; // member  
o friend bool operator==(const Height &r, const Height &r);  
Add the following test cases in your driver and add code to label height and new line for  
formatting as needed:

Height h6(0, 5); // feet: 0, inches: 5  
// Print height (add code to label height and new line as needed)  
h6.print(); // h6: 0’ 5”  
t6.increment(12); // ignore, feet: 0, inches: 5  
t6.increment(-3); // ignore, feet: 0, inches: 5  
t6.increment(); // feet: 0, inches: 6  
t6.increment(10); // feet: 1, inches: 4  
cout << “Total inches: “ << h6.totalInches() << endl; // 16  
// Print height (add code to label height and new line as needed)  
h6.print(); // h6: 1’ 4”  
if (h3 == h6)  
cout << “h3 is the same as h6” << endl;  
else  
cout << “h3 is not the same as h6” << endl;  
// should output: h3 is not the same as h6

Source code below:

HeightV2.cpp:

/\* Program: Height Class Version 2

Author: Ean Zheng

Class : CSCI 140

Date : 4/20/2025

Description :

I certify that the code below is my own work.

Exception(s) : N/A

\*/

#include <iostream>

using namespace std;

#include "Height.h"

int main() {

// Create 2 Height objects

Height h1; // feet: 0, inches: 0

Height h2(5); // feet: 5, inches: 0

// Create some Height objects (same as original version)

Height h3(5, 8); // feet: 5, inches: 8

Height h4(-1, 5); // feet: 0, inches: 5 (invalid feet so set to 0)

Height h5(6, 15); // feet: 6, inches: 0 (invalid inches so set to 0)

// Print height h3

Height array[]{ h1, h2, h3, h4, h5 };

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

array[i].print();

}

cout << "h3: ";

h3.print(); // h3: 5’ 8”

cout << endl;

// Add more code below to print h4 and h5 like h3 above

// Perform various operations

h3.setFeet(-2); // feet: 5, inches: 8, feet stay the same

h3.setInches(10); // feet: 5, inches: 10

cout << "feet: " << h3.getFeet() << ", inches: " << h3.getInches() << endl; // 5 10

h4.setFeet(6); // feet: 6, inches: 5

h4.setInches(12); // feet: 6, inches: 5, inches stay the same

cout << "feet: " << h4.getFeet() << ", inches: " << h4.getInches() << endl; // 6 5

h5.setInches(10); // feet: 6, inches: 10

h5.increment(); // feet: 6, inches: 11

h5.increment(); // feet: 7, inches: 0

cout << "h5: ";

h5.print(); // h3: 7’ 0”

cout << endl;

//Add more test cases if needed

Height h6(0, 5); // feet: 0, inches: 5

// Print height (add code to label height and new line as needed)

cout << "h6: ";

h6.print(); // h6: 0’ 5”

h6.increment(12); // ignore, feet: 0, inches: 5

h6.increment(-3); // ignore, feet: 0, inches: 5

h6.increment(); // feet: 0, inches: 6

h6.increment(10); // feet: 1, inches: 4

cout << "Total inches : " << h6.totalInches() << endl; // 16

// Print height (add code to label height and new line as needed)

cout << "h6: ";

h6.print(); // h6: 1’ 4”

if (h3 == h6)

cout << "h3 is the same as h6" << endl;

else

cout << "h3 is not the same as h6" << endl;

// should output: h3 is not the same as h6

}

Height.cpp:

/\* Program: Height Class Version 2

Author: Ean Zheng

Class : CSCI 140

Date : 4/20/2025

Description :

I certify that the code below is my own work.

Exception(s) : N/A

\*/

#include <iostream>

using namespace std;

#include "Height.h"

Height::Height(int f, int i) : feet(f), inches(i) {

if (feet < 0)feet = 0;

if (inches < 0 || inches > 11) inches = 0;

}

void Height::setFeet(int f) {

if (f >= 0)feet = f;

}

void Height::setInches(int i) {

if (i >= 0 && i <= 11) inches = i;

}

int Height::getFeet() const {

return feet;

}

int Height::getInches() const {

return inches;

}

void Height::print() const {

cout << feet << "' " << inches << "\"" << endl;

}

void Height::increment() {

++inches;

if (inches == 12) {

inches = 0;

++feet;

}

}

void Height::increment(int inches) {

if (inches > 0 && inches < 12) {

this->inches = this->inches + inches;

if (this->inches >= 12) {

this->inches -= 12;

++feet;

}

}

}

int Height::totalInches() const {

return (feet \* 12) + inches;

}

bool Height::operator==(const Height& r) const {

if (feet == r.getFeet() && inches == r.getInches())

return true;

else if (feet != r.getFeet() || inches != r.getInches())

return false;

}

Height.h:

/\* Program: Height Class Version 2

Author: Ean Zheng

Class : CSCI 140

Date : 4/20/2025

Description :

I certify that the code below is my own work.

Exception(s) : N/A

\*/

#ifndef HEIGHT\_H

#define HEIGHT\_H

#include <string>

using namespace std;

class Height {

private:

int feet;

int inches;

public:

Height(int f = 0, int i = 0);

void setFeet(int f = 0);

void setInches(int i = 0);

int getFeet() const;

int getInches() const;

void print() const;

void increment();

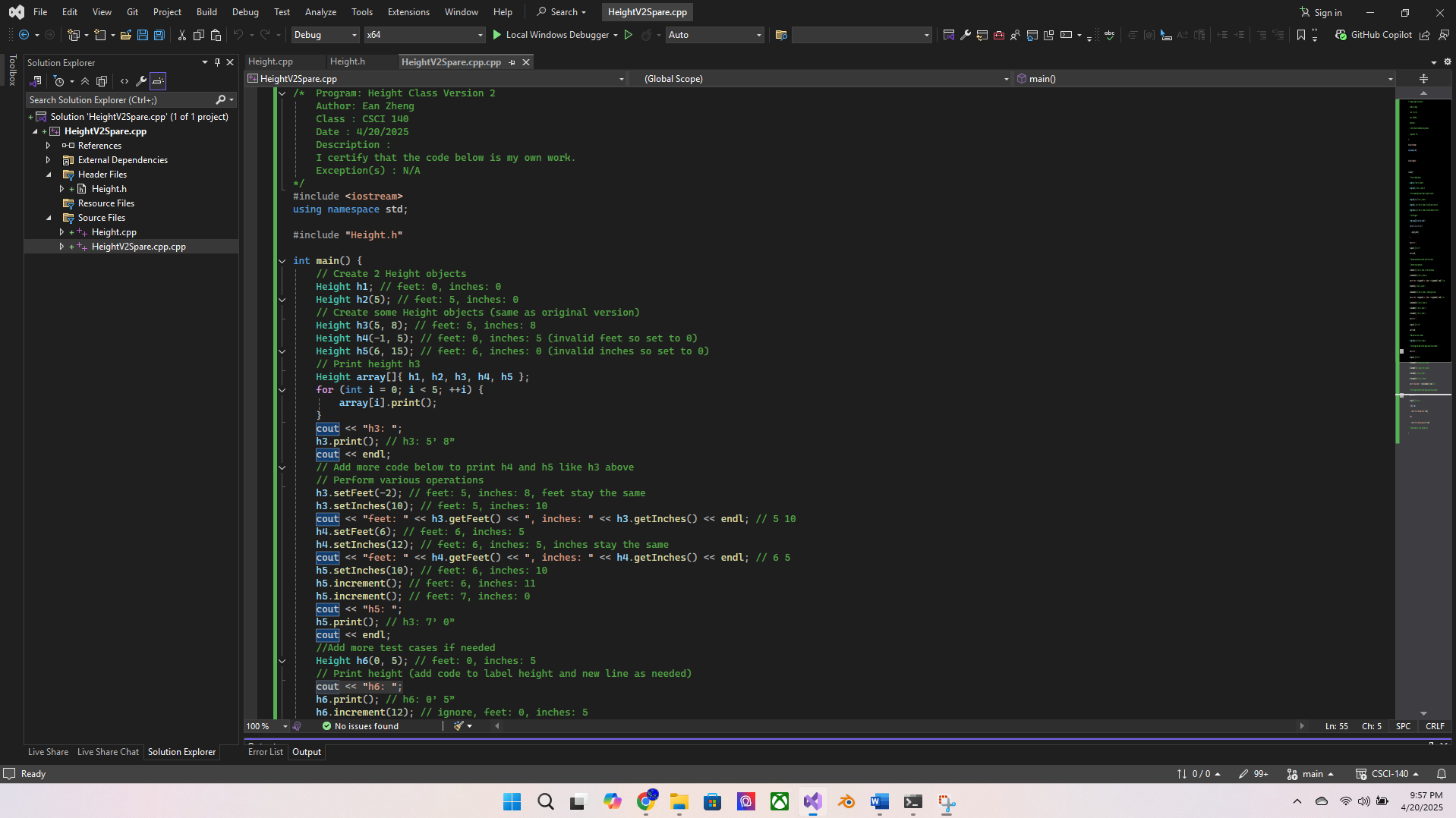
void increment(int inches);

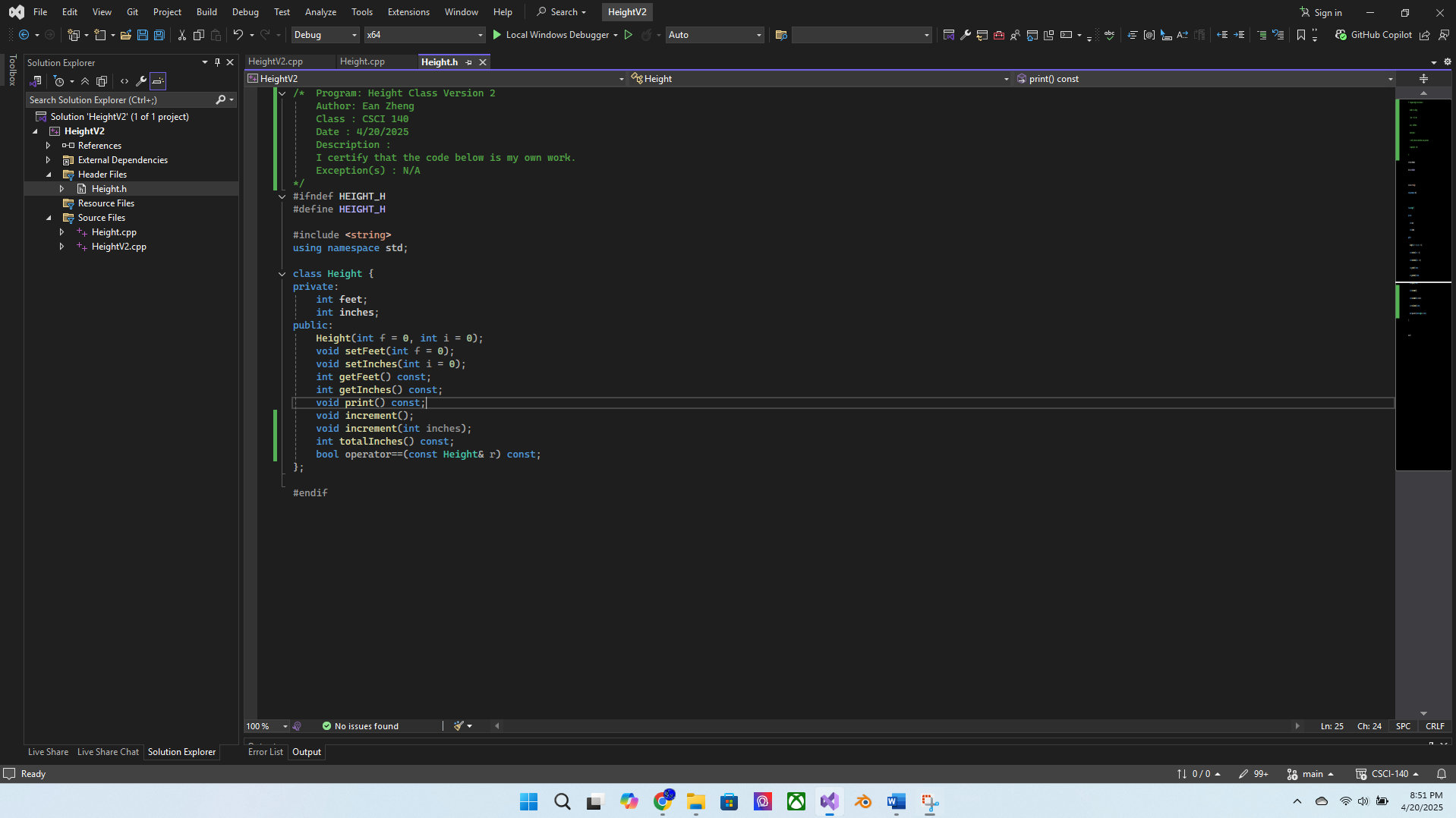
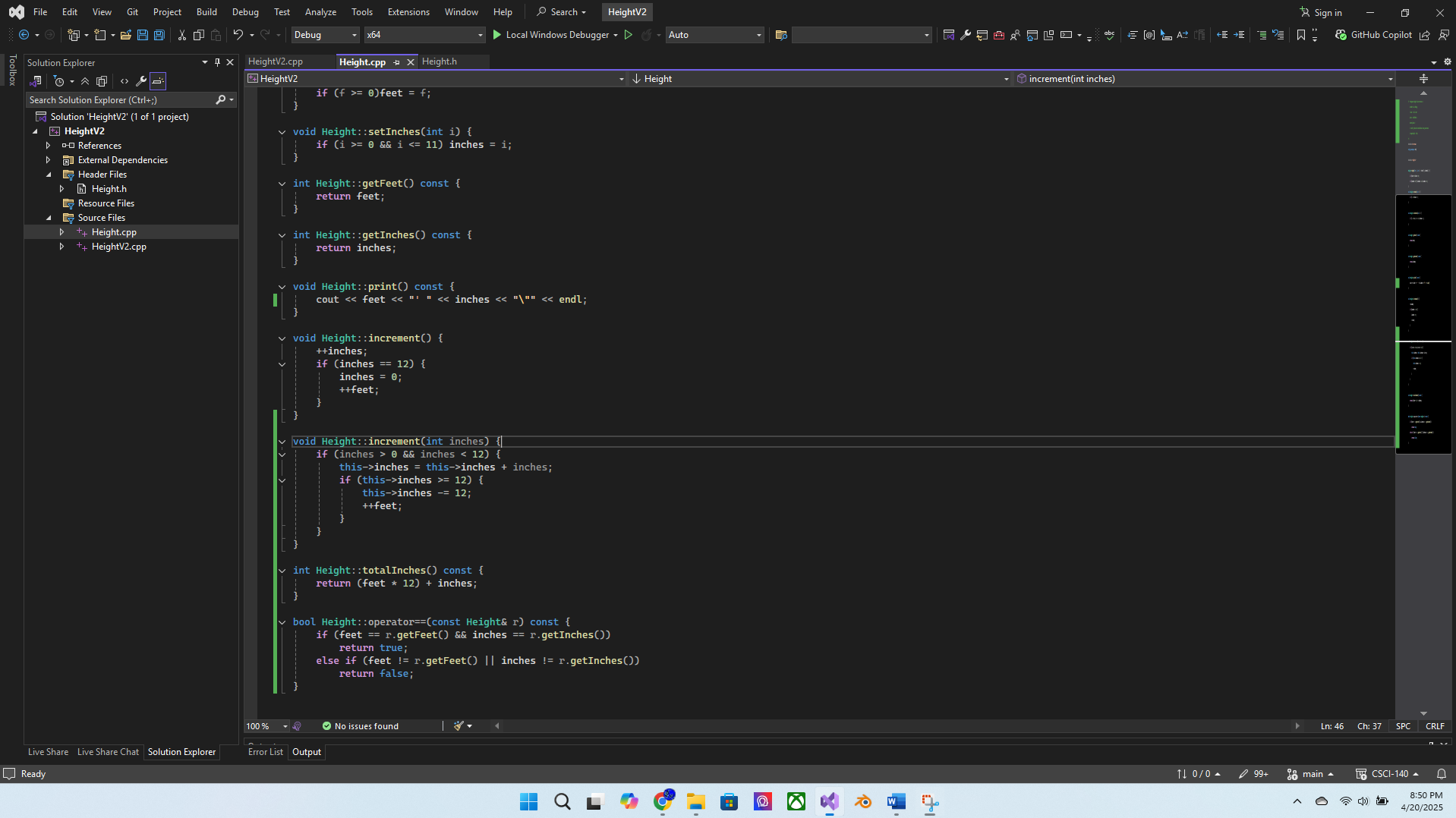
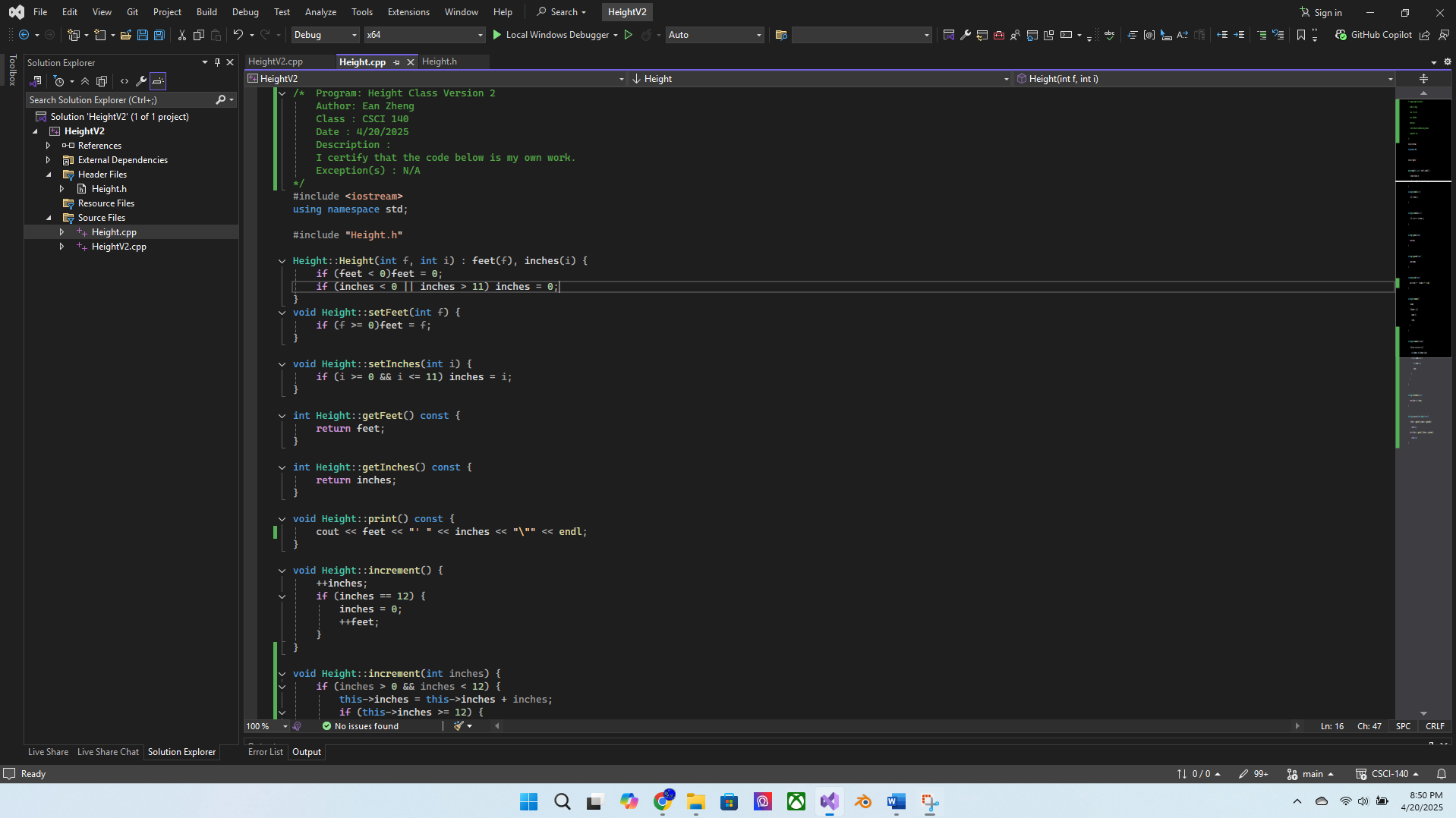
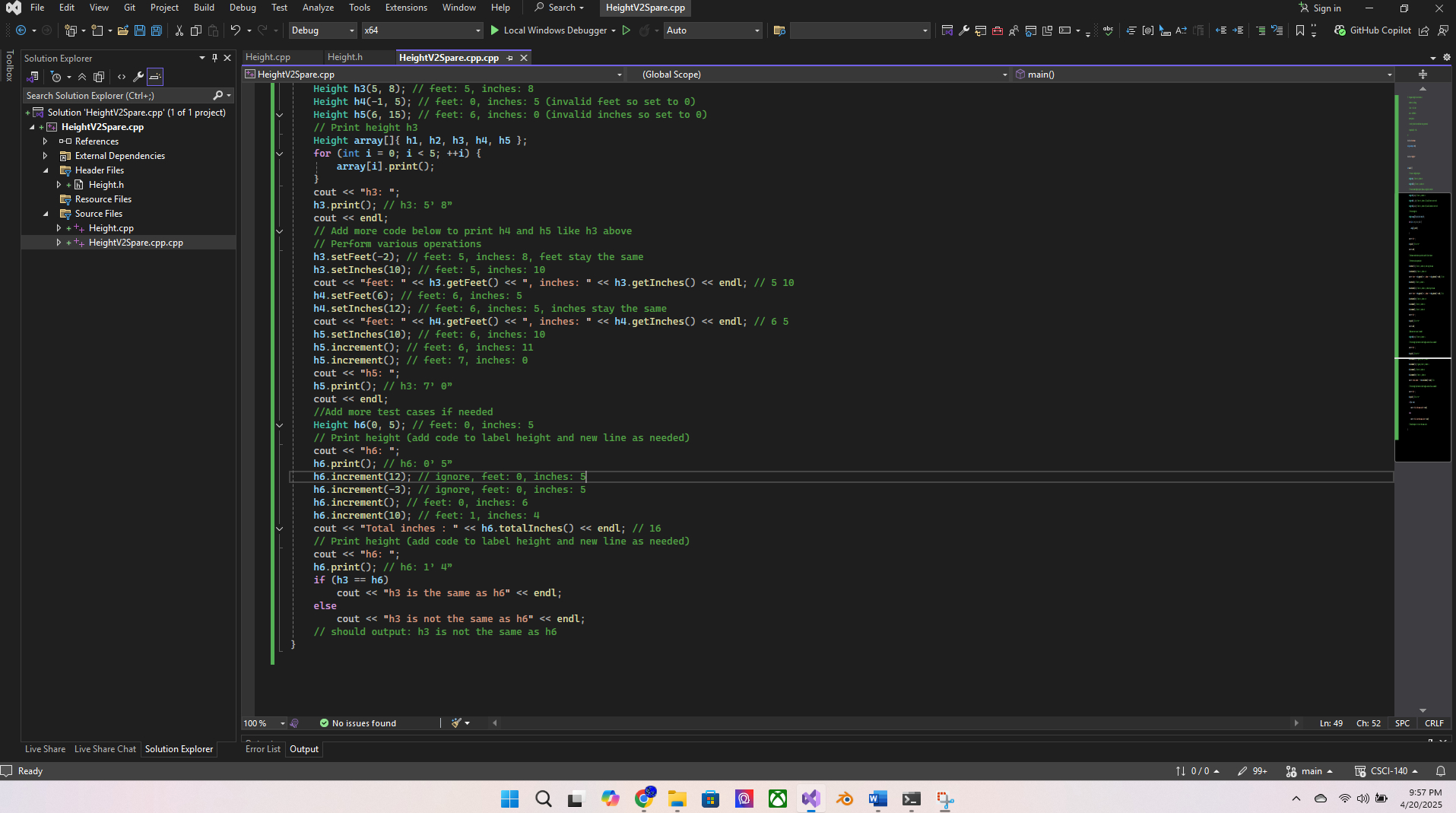
int totalInches() const;

bool operator==(const Height& r) const;

};

#endif





Input/output below:

0' 0"

5' 0"

5' 8"

0' 5"

6' 0"

h3: 5' 8"

feet: 5, inches: 10

feet: 6, inches: 5

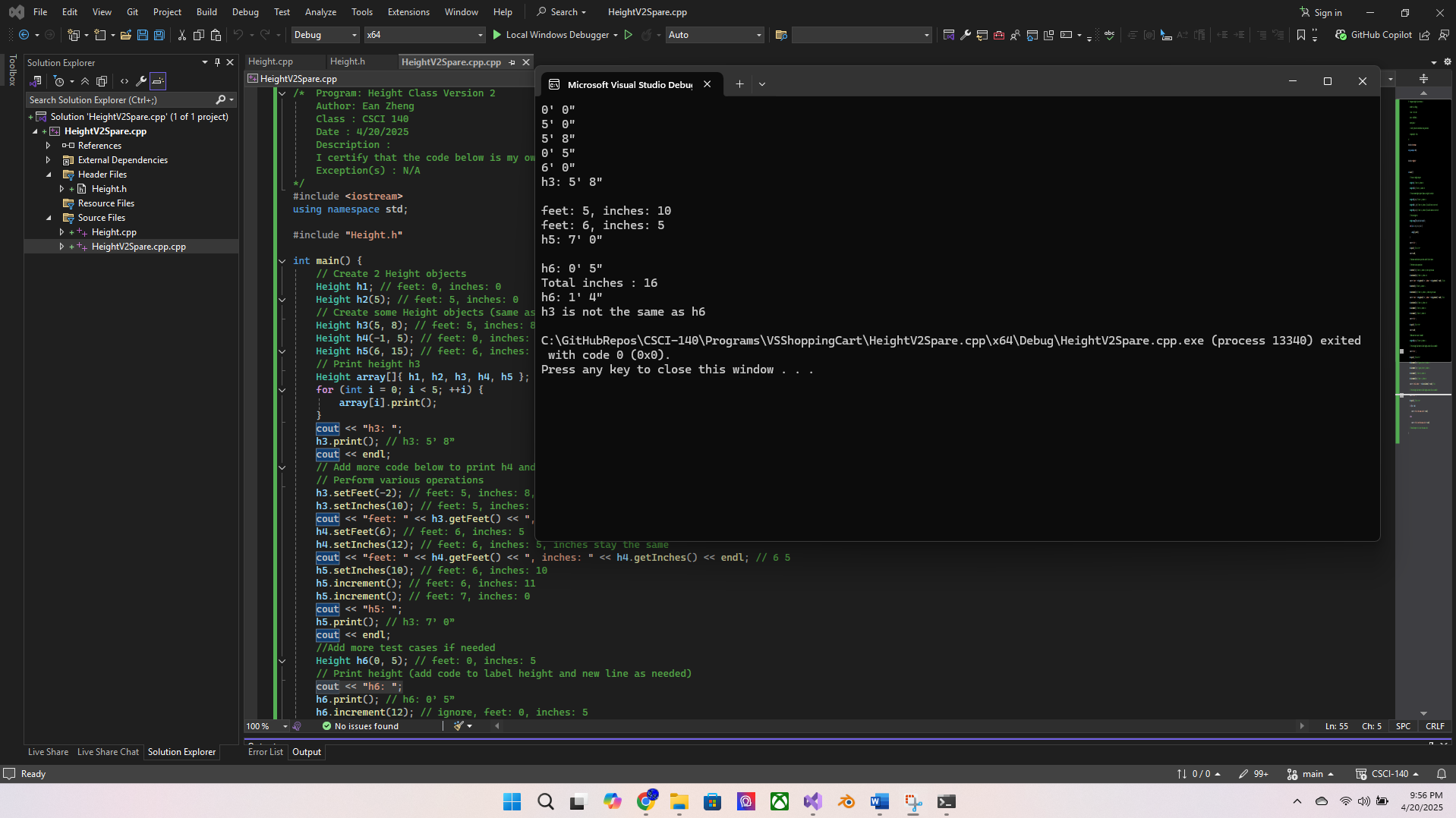
h5: 7' 0"

h6: 0' 5"

Total inches : 16

h6: 1' 4"

h3 is not the same as h6



Question 1: It is possible to use a single int private variable totInches that represents  
height in inches (instead of variables feet and inches) to implement the Height class  
without changing the interface (i.e., public member functions stay the same and the  
application still works the same way). Explain how you would go about doing that.

Yes. I would implement that by adding that every time a new feet or inch value is assigned or upon object initialization, the total Inches value is recalculated given the current/new feet and inches variables. If the increment function is called, the only change would be to add the increment to the variable.

Question 2: List some good reasons for overloading operators in a class.

To reduce work, reduce amount of code, to make code simpler and easier to understand, and make writing the code easier and more convenient.

Extra Credit (2 points): Modify your Height class to add the following operators: +  
and -. You can add or subtract (absolute difference), and the result would be a Height  
object. Given two Height objects, h3 and h6, you can do:  
// assume h3 (5’ 8”) and h6 (1’ 4”)  
Height h7 = h3 + h6;  
Height h8 = h3 - h6;  
Height h9 = h6 – h3;  
// Print height (add code to label height and new line as needed)  
h7.print(); // h7: 7’ 0”  
// Print height (add code to label height and new line as needed)  
h8.print(); // h8: 4’ 4”  
// Print height (add code to label height and new line as needed)  
h9.print(); // h9: 4’ 4”  
// add more test cases as needed

Source code below:

HeightV2.cpp:

/\* Program: Height Class Version 2 Extra Credit Ver.

Author: Ean Zheng

Class : CSCI 140

Date : 4/20/2025

Description :

I certify that the code below is my own work.

Exception(s) : N/A

\*/

#include <iostream>

using namespace std;

#include "Height.h"

int main() {

// Create 2 Height objects

Height h1; // feet: 0, inches: 0

Height h2(5); // feet: 5, inches: 0

// Create some Height objects (same as original version)

Height h3(5, 8); // feet: 5, inches: 8

Height h4(-1, 5); // feet: 0, inches: 5 (invalid feet so set to 0)

Height h5(6, 15); // feet: 6, inches: 0 (invalid inches so set to 0)

// Print height h3

Height array[]{ h1, h2, h3, h4, h5 };

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

array[i].print();

}

cout << "h3: ";

h3.print(); // h3: 5’ 8”

cout << endl;

// Add more code below to print h4 and h5 like h3 above

// Perform various operations

h3.setFeet(-2); // feet: 5, inches: 8, feet stay the same

h3.setInches(10); // feet: 5, inches: 10

cout << "feet: " << h3.getFeet() << ", inches: " << h3.getInches() << endl; // 5 10

h4.setFeet(6); // feet: 6, inches: 5

h4.setInches(12); // feet: 6, inches: 5, inches stay the same

cout << "feet: " << h4.getFeet() << ", inches: " << h4.getInches() << endl; // 6 5

h5.setInches(10); // feet: 6, inches: 10

h5.increment(); // feet: 6, inches: 11

h5.increment(); // feet: 7, inches: 0

cout << "h5: ";

h5.print(); // h3: 7’ 0”

cout << endl;

//Add more test cases if needed

Height h6(0, 5); // feet: 0, inches: 5

// Print height (add code to label height and new line as needed)

cout << "h6: ";

h6.print(); // h6: 0’ 5”

h6.increment(12); // ignore, feet: 0, inches: 5

h6.increment(-3); // ignore, feet: 0, inches: 5

h6.increment(); // feet: 0, inches: 6

h6.increment(10); // feet: 1, inches: 4

cout << "Total inches : " << h6.totalInches() << endl; // 16

// Print height (add code to label height and new line as needed)

cout << "h6: ";

h6.print(); // h6: 1’ 4”

if (h3 == h6)

cout << "h3 is the same as h6" << endl;

else

cout << "h3 is not the same as h6" << endl;

// should output: h3 is not the same as h6

// assume h3 (5’ 8”) and h6 (1’ 4”)

h3.setInches(8);

Height h7 = h3 + h6;

Height h8 = h3 - h6;

Height h9 = h6 - h3;

// Print height (add code to label height and new line as needed)

cout << "h7: ";

h7.print(); // h7: 7’ 0”

// Print height (add code to label height and new line as needed)

cout << "h8: ";

h8.print(); // h8: 4’ 4”

// Print height (add code to label height and new line as needed)

cout << "h9: ";

h9.print(); // h9: 4’ 4”

// add more test cases as needed

}

Height.cpp:

/\* Program: Height Class Version 2 Extra Credit Ver.

Author: Ean Zheng

Class : CSCI 140

Date : 4/20/2025

Description :

I certify that the code below is my own work.

Exception(s) : N/A

\*/

#include <iostream>

using namespace std;

#include "Height.h"

Height::Height(int f, int i) : feet(f), inches(i) {

if (feet < 0)feet = 0;

if (inches < 0 || inches > 11) inches = 0;

}

void Height::setFeet(int f) {

if (f >= 0)feet = f;

}

void Height::setInches(int i) {

if (i >= 0 && i <= 11) inches = i;

}

int Height::getFeet() const {

return feet;

}

int Height::getInches() const {

return inches;

}

void Height::print() const {

cout << feet << "' " << inches << "\"" << endl;

}

void Height::increment() {

++inches;

if (inches == 12) {

inches = 0;

++feet;

}

}

void Height::increment(int inches) {

if (inches > 0 && inches < 12) {

this->inches = this->inches + inches;

if (this->inches >= 12) {

this->inches -= 12;

++feet;

}

}

}

int Height::totalInches() const {

return (feet \* 12) + inches;

}

bool Height::operator==(const Height& r) const {

if (feet == r.getFeet() && inches == r.getInches())

return true;

else if (feet != r.getFeet() || inches != r.getInches())

return false;

}

Height Height::operator+(Height rhs) {

int tempfeet = feet + rhs.getFeet();

int tempinches = inches + rhs.getInches();

if (tempinches >= 12) {

tempinches -= 12;

++tempfeet;

}

Height result(tempfeet, tempinches);

return result;

}

Height Height::operator-(Height rhs) {

int tempfeet = max(feet, rhs.getFeet()) - min(feet, rhs.getFeet());

int tempinches = max(inches, rhs.getInches()) - min(inches, rhs.getInches());

if (tempinches < 0) {

tempinches = 12 + tempinches;

--tempfeet;

}

Height result(tempfeet, tempinches);

return result;

}

Height.h:

/\* Program: Height Class Version 2 Extra Credit Ver.

Author: Ean Zheng

Class : CSCI 140

Date : 4/20/2025

Description :

I certify that the code below is my own work.

Exception(s) : N/A

\*/

#ifndef HEIGHT\_H

#define HEIGHT\_H

#include <string>

using namespace std;

class Height {

private:

int feet;

int inches;

public:

Height(int f = 0, int i = 0);

void setFeet(int f = 0);

void setInches(int i = 0);

int getFeet() const;

int getInches() const;

void print() const;

void increment();

void increment(int inches);

int totalInches() const;

bool operator==(const Height& r) const;

Height operator+(Height rhs);

Height operator-(Height rhs);

};

#endif

Input/output below:

0' 0"

5' 0"

5' 8"

0' 5"

6' 0"

h3: 5' 8"

feet: 5, inches: 10

feet: 6, inches: 5

h5: 7' 0"

h6: 0' 5"

Total inches : 16

h6: 1' 4"

h3 is not the same as h6

h7: 7' 0"

h8: 4' 4"

h9: 4' 4"

