Module 9:

Operator Overloading, Friends and References

COP2274

In-class Assignments

UNIVERSITY of FLORIDA

M9A Operator overloading (non-member operator)

1. Write a class called **Sandwich** with *private member variables* for a weight(double) and a size(char).
2. Write ***a default constructor*** that initializes the weight as 50 grams and the size as ‘S’.
3. Write ***a custom constructor*** that takes in just a weight and initialize the member variable for weight with it. In ***the custom constructor***, initialize a size (‘S’ for small, ‘M’ for medium, ‘L’ for large, or ‘X’ for extra large) using the ranges of the weight (0 <= S < 150, 150 <= M < 300, 300 <= L < 500, 500 <= X).

M9A Operator overloading (non-member operator)

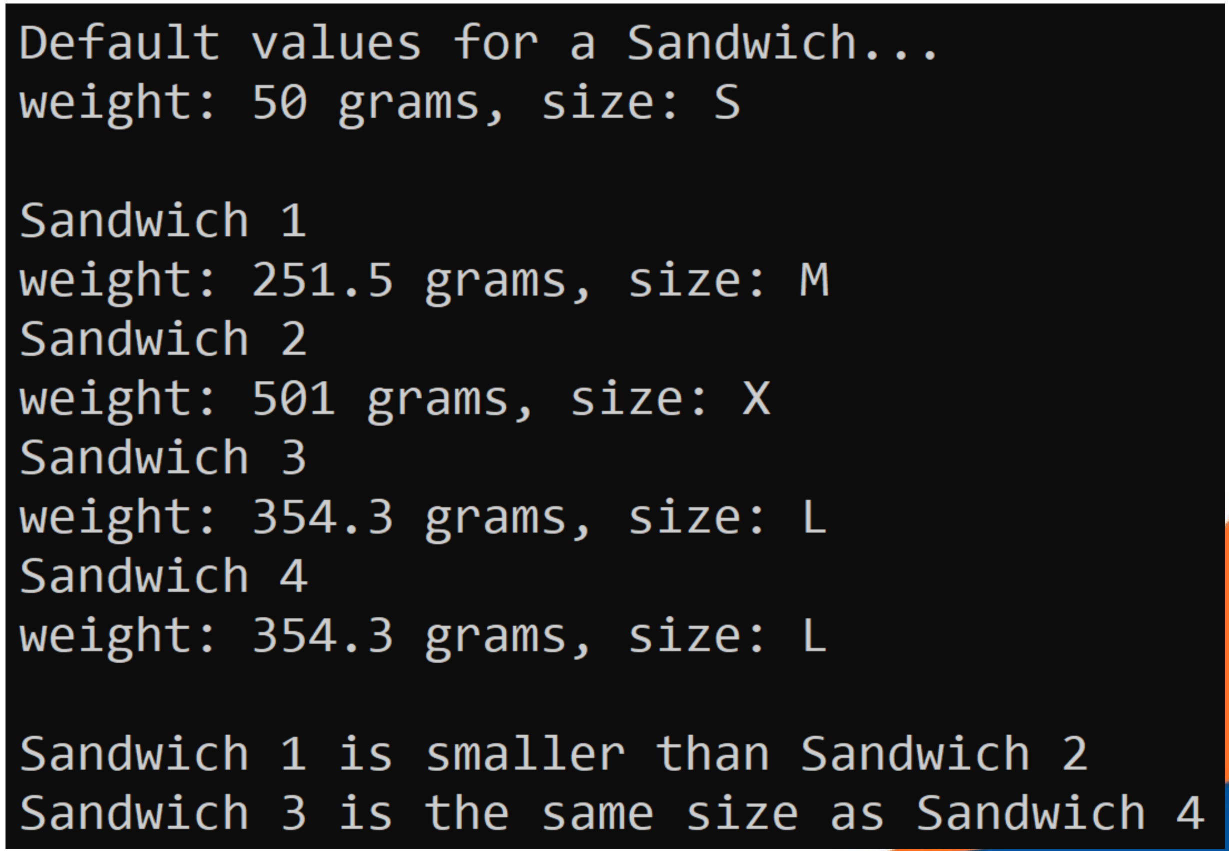
4. Write a member function called print() that prints out according to the test case.
5. Overload **>** operator and **==** operator for **Sandwich** (as non-member functions) to return true or false by comparing the weight.
6. In your main(), declare several **Sandwich** objects that are initialized by the default constructor or the custom constructor and then call print() on the **Sandwich** objects to print the values of the member variables and then, compare the **Sandwich** objects with your overloaded operators (**>**, **==**) as shown in the test case.

M9A Operator overloading (non-member operator)

Note:

- *You can declare **Sandwich** objects with hard-coded values as shown in the test case.*

Test case



```
Default values for a Sandwich...
weight: 50 grams, size: S

Sandwich 1
weight: 251.5 grams, size: M
Sandwich 2
weight: 501 grams, size: X
Sandwich 3
weight: 354.3 grams, size: L
Sandwich 4
weight: 354.3 grams, size: L

Sandwich 1 is smaller than Sandwich 2
Sandwich 3 is the same size as Sandwich 4
```

M9B Operator overloading (non-member friend operator)

1. Write a class called **Point** that contains two <u>*private*</u> member variables for x and y coordinates (double).
2. Write ***a default constructor*** that initializes the member variables to 0. Define ***the default constructor*** with an initialization section.
3. Write ***a custom constructor*** with two parameters that initializes each member variable with its corresponding parameter. Define ***the custom constructor*** with an initialization section.

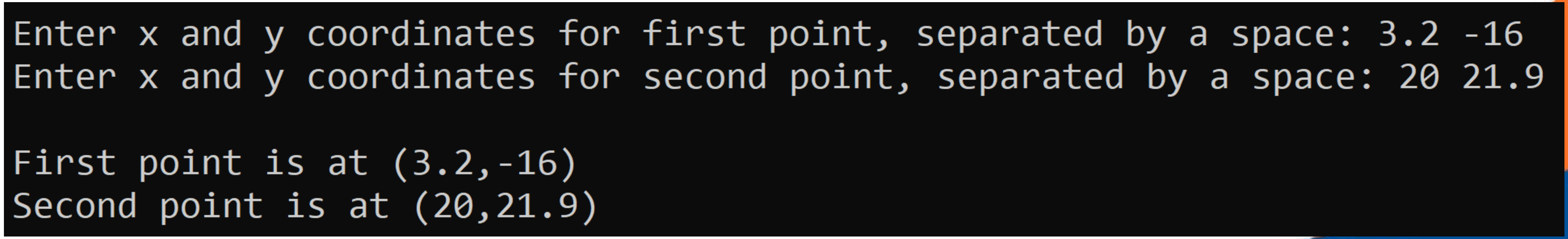
M9B Operator overloading (non-member friend operator)

4. Overload **<<** operator for **Point** (as a non-member friend function). The overloaded **<<** operator returns an **ostream** object.
   Hint: The prototype of the overloaded operator << is as follows.
   friend ostream& operator << (ostream& output, const Point& p);
5. Overload **>>** operator for **Point** (as a non-member friend function). The overloaded **>>** operator returns an **istream** object.
   Hint: The prototype of the overloaded operator >> is as follows.
   friend istream& operator >> (istream& input, Point& p);

M9B Operator overloading (non-member friend operator)

6. In your main(), declare two **Point** objects that are initialized by the default constructor. Prompt user inputs as shown in the test case and save the user inputs with your overloaded **>>** operator. And display the x and y coordinates of the user inputs with your overloaded **<<** operator as shown in the test case.

Test case



```
Enter x and y coordinates for first point, separated by a space: 3.2 -16
Enter x and y coordinates for second point, separated by a space: 20 21.9

First point is at (3.2,-16)
Second point is at (20,21.9)
```

M9C Operator overloading (member operator)

1. Write a class called **PrimeNumber** with a *private member variable* to store a prime number.
2. Write ***a default constructor*** that initializes the member variable to 2 and displays "The default prime number is 2" as shown in the test case. Define ***the default constructor*** with an initialization section.
3. Write ***a custom constructor*** that takes in a parameter and initializes the member variable with it. Define ***the custom constructor*** with an initialization section.
4. Write a constant accessor function for the member variable.

M9C Operator overloading (member operator)

5. Write a constant member function called *isPrime()* that takes in an integer and returns true if it is prime or false if it is not.
6. Overload the postfix unary **++** operator (as a member function), which returns a constant **PrimeNumber** object. You may use the member *isPrime()* function here, which will increment the member variable until *isPrime()* returns true.
7. In your main(), declare a **PrimeNumber** object that is initialized by the default constructor.

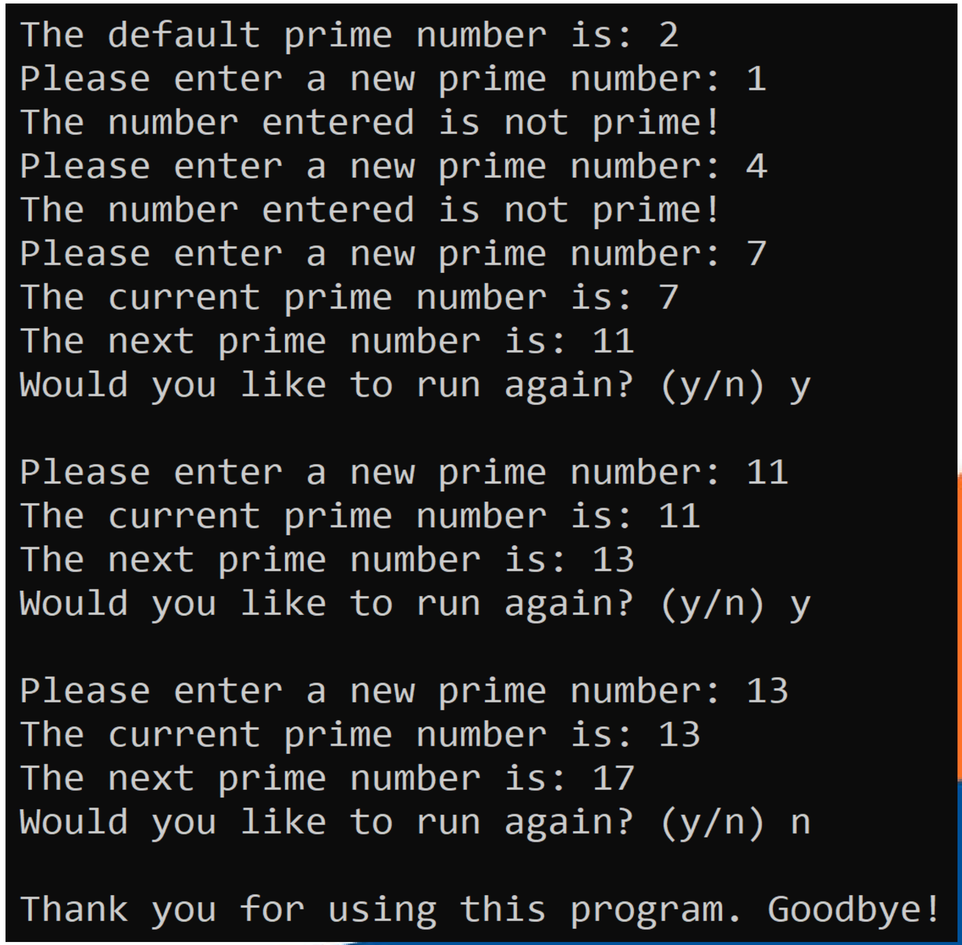
M9C Operator overloading (member operator)

8. Then prompt the user to enter a new prime number. Continue prompting the user for a new number until the number they enter is prime, as shown in test case. After the number is determined to be prime, reinitialize the existing **PrimeNumber** object using the anonymous object created by calling the custom constructor with the new prime number (or user input). Display the new prime number by calling the accessor function as shown in the test case.

M9C Operator overloading (member operator)

8. Finally, create a new **PrimeNumber** object that is initialized by pre-increamenting the first **PrimeNumber** object. Display your results as shown in the test case, and prompt the user if they would like to enter another number (y/n).

Test case



```
The default prime number is: 2
Please enter a new prime number: 1
The number entered is not prime!
Please enter a new prime number: 4
The number entered is not prime!
Please enter a new prime number: 7
The current prime number is: 7
The next prime number is: 11
Would you like to run again? (y/n) y

Please enter a new prime number: 11
The current prime number is: 11
The next prime number is: 13
Would you like to run again? (y/n) y

Please enter a new prime number: 13
The current prime number is: 13
The next prime number is: 17
Would you like to run again? (y/n) n

Thank you for using this program. Goodbye!
```