EECE 1080C/CS1021C - Laboratory 8 - Summer 2019

Class Practice

Topics covered: Classes & Functions

Due:

Sunday July 14, 2019 by 11:59PM

Objective:

This assignment is an introduction to the C++ classes. We will learn the basic of class design, and we will put it to use creating a couple of more advanced classes.

Collaboration:

As with most laboratory assignments in this course this laboratory assignment is to performed by an individual student. You can help each other learn by reviewing assignment materials, describing to each other how you are approaching the problem, and helping each other with syntax errors. You can get help from tutors, teaching assistants, and instructors. Having similar code for some assignments is expected but most assignments have multiple different solutions. Your code is expected to be different.

Please document any help you receive from other students, teaching assistants, or instructors. Just add the names to the top of your source file.

Having someone else code for you, sharing code with other students, or copy-pasting code from the internet or previous terms , is cheating. A helper should "teach you to fish, not feed you the fish". Laboratory assignments prepare you for exams so be smart.

Highlights:

- Please review the chapter on classes.
- Please read the specification below carefully.
- Review **Grading Rubric** at the end of this document.
- Please ask appropriate questions when necessary.
- Remember to download the starter code for each part.
- Finish Parts A and B during the laboratory session.
    - Parts A and B are your milestone for this week
    - Parts A and B will be graded during recitation this week.
    - You will want to start working on at least Part A before you get to the recitation session.
- All parts should be submitted to blackboard when complete using the names and folder (lab8) indicated in this document.

EECE 1080C/CS1021C - Laboratory 8 - Summer 2019

Class Practice

Topics covered: Classes & Functions

Specification:

Part A: Class Practice - Warm Up Exercises

Problem Statement:

In part A, you will create two programs to solve for surface area, and volume of a specific shape. A list of geometric shapes can be found on the following:

http://www.mathsisfun.com/geometry/common-3d-shapes.html

To get you started, review the sample_shape.cpp for sample code of a 2D rectangular object. Make sure you view it in "raw" mode.

Task 1: Pick a 3D shape from the ***mathisfun*** website.

Source File: lab8/shape1.cpp

Using the example solution given above as your guide, create a solution to solve for the surface area and volume of your chosen shape using C++ classes.

Provide sample calculations for this shape to verify that it works properly.

Please create proper test cases for this in main() to verify proper functioning.

Task 2: Pick another 3D shape from the ***mathisfun*** website.

Source File: lab8/shape2.cpp

Using the program developed in ***Task 1*** as a template, develop one more C++ class contained in a separate C++ program to solve for surface area and volume of your selected object.

Provide sample calculations and sample output for each of these shapes.

Please create proper test cases for this in main() to verify proper functioning.

EECE 1080C/CS1021C - Laboratory 8 - Summer 2019

Class Practice

Topics covered: Classes & Functions

Part B: Create Point Class - Milestone One
Source File: lab8/point_class.cpp

Problem Statement:

Start by downloading the point_class.cpp file on blackboard for the ***Point*** class. It implements operator overloading so you can use cin/cout with point objects.

Complete the provided "Point" class which allows a programmer to store an x, y coordinate pair. It should have at least two constructors, at least one set function, and get functions for x and y.

The overloaded I/O (cin/cout) functions have been provided for your use.

Please create proper test cases for this in main() to verify proper functioning.

Part C: Line Class
Source File: lab8/line_class.cpp

Problem Statement:
Start by downloading the line_class.cpp from blackboard for the ***Line*** class. It implements operator overloading so you can use cin/cout with line objects.

A line will be made up of two points.

Create an object to implement a "line" class which allows the programmer to store a line. This class must use the "point" class developed in Part B. The object should have two constructors, appropriate set/get functions, and overloaded I/O (cin/cout) functions. It should include functions the return the proper value for the following:

- Determine the slope of a line
- Determine the length of a line
- Determine the y-intercept of a line
- Determine if the line is vertical
- Determine if the line is horizontal
- Determine if the line is parallel to another line
-

Please create proper test cases for this in main() to verify proper functioning.

EECE 1080C/CS1021C - Laboratory 8 - Summer 2019

Class Practice

Topics covered: Classes & Functions

Note: You should make it easy on yourself and use four integers representing the two ends of the line for the second constructor

Part D: Let's Practice Strings with classes

Source File: lab8/string_manip.cpp

Start by downloading the string_manip.cpp for the ***stringManip*** class.

Please review the "cctype" and string review given on the last page of this document.

It will give you a shell to implement the following class prototype:

```
class stringManip {
    public:
      stringManip();
      stringManip(string input);
      string retrieve();
      void setString(string input);
      void chop();
      void padString(int n);
      void center(int length);
      void truncate(int n);
      void removeNonAlpha();
      void convertToUpperCase();
      void convertToLowerCase();
    private:
      string tobeEdited;
};
```

All functions act on the ***tobeEdited*** string inside of the class unless indicated otherwise.

Please implement the functions indicated plus the constructors, and set/get (retrieve and setString) functions.

Function/method descriptions.

1. chop() - remove both leading and trailing spaces from the string.

2. padString(int n) - Write a function to make sure that the string is at least "n" bytes in length. Please add a space or spaces Add blanks toAdd blanks toto the end of the string to accomplish this. If the

string length is already equal to or larger than "n", do not modify the string.

3. center(int length) - Write a function to center the string within the space specified by length. You should remove any existing leading and trailing spaces in the string before centering it.

   Add blanks to the front and back of the string. If an odd number of blanks have to be added, put the extra blank on the right. You should store the result back in the "tobeEdited" string.

4. truncate(int n) is a function which shortens the string to n characters.

   If the string is already shorter than n, the function should not change the string. The characters should be removed from the end of the string.

5. removeNonAlpha() is function that removes all characters (including spaces) that are not alphabetical from the string. The isalpha function may be helpful for this purpose.

6. convertToUpperCase() is a function that converts all lowercase characters in the string to uppercase. All other characters remain the same.
7. convertToLowerCase() is a function that converts all lowercase characters in the string to uppercase. All other characters remain the same.

Assignment Grading Rubric:

Program Grading Rubric:

- 25% - Two shapes from part A work properly
- 25% - Point class works properly
- 25% - Line class works properly
- 25% - stringManip implementation

To receive full credit for this assignment your program should follow these guidelines:

- Milestone signed off during recitation session
- Implement all .cpp files
- Assignment uploaded to blackboard

EECE 1080C/CS1021C - Laboratory 8 - Summer 2019

Class Practice

Topics covered: Classes & Functions

- Only use global variables for constants
- Appropriate use of classes and/or functions
- You must format the code properly using proper indentation, descriptive variable and function names, and proper commenting.

CCTYPE Review

cctype is library containing a set of functions that can classify or transform characters. ccytpe can be used in your program by using #include <cctype>.

Character Testing in cctype:
Few functions that can classify a character are shown below. These are also called "character testing" functions (hence, **bool** return-type functions).

Assume that we defined a char **ch,** the character testing functions, and their behaviour is shown below:

Function	**Call**	**Function return**
isalpha	isalpha(ch)	Returns true if **ch** is a letter, false otherwise.
isalnum	isalnum(ch)	Returns true if **ch** is a letter or digit, false otherwise
isdigit	isdigit(ch)	Returns true if **ch** is a digit between 0-9, false otherwise
islower	islower(ch)	Returns true if **ch** is lowercase letter, false otherwise
isprint	isprint(ch)	Returns true if **ch** is a printable character, false otherwise
ispunct	ispunct(ch)	Returns true if **ch** is a punctuation character, false otherwise
isupper	isupper(ch)	Returns true if **ch** is an uppercase letter, false otherwise
isspace	isspace(ch)	Returns true if **ch** is a whitespace character, false otherwise

Character case-conversion in cctype:
These are cctype functions that output the opposite case equivalent of a character. They **do not** modify the original character itself. If **ch** is defined as a character, character case-conversion functions are shown below.

Function	**Call**	**Function output**

EECE 1080C/CS1021C - Laboratory 8 - Summer 2019

Class Practice

Topics covered: Classes & Functions

tolower	tolower(ch)	if **ch** is uppercase letter, return lowercase equivalent; otherwise, return input **unchanged.**
toupper	toupper(ch)	if **ch** is lowercase letter, return uppercase equivalent; otherwise, return input **unchanged.**

See usage examples on the next page.
Eg:-

```
char ch1 = 'H';
char ch2 = 'e';
char ch3 = '!';

cout << toupper(ch1);  // displays 'H'
cout << toupper(ch2);  // displays 'E'
cout << toupper(ch3);  // displays '!'
cout << tolower(ch1);  // displays 'h'
```