CSIS 135 – Programming in C/C++
Spring 2017

Homework #7 (LAST ONE) – Due Date: *06/12/17 at 7:50 pm*

NO LATES WILL BE ACCEPTED ON THIS HOMEWORK

For this homework ONLY – you will SUBMIT FILES ON MOODLE

IMPORTANT INSTRUCTIONS (follow these to avoid losing points – NO LATES)

- Submit your source code files (all .h and .cpp files) and your UML diagrams ON MOODLE - no hypergrade for this assignment. I'll go over how to submit on Moodle in class – ask me if you are unsure.
- ***You will email three files for each program (a .h file, a .cpp file for the implementation and a .cpp file for the client code).***
- Make sure your name is on each file you submit.
- **Make sure and use three separate files for your object oriented programs (this means for each program you need to submit three files on Moodle (two .cpp files and 1 .h file) plus UML diagrams. Do not use inline functions.** ***IF YOU VIOLATE THESE RULES – points off.***
- **NOTE:** These programs require UML diagrams instead of pseudocode. This is a LONG homework with 3 programs…the reason for the length is to give you extra practice for the final.
- ***SINCE THIS HW IS DUE THE NIGHT BEFORE YOUR FINAL AND THE LAST DAY OF REGULAR CLASSES NO LATE ASSIGNMENTS WILL BE ACCEPTED***
  - ***NO EXCEPTIONS – do not email and ask for one***
  - ***Turn in whatever you have completed on the due date***

Homework #7 - Program 1: Employee Class – Based on Chapter 10 Lecture

Write a class named *Employee* that has the following member variables:

- *name*: The name attribute holds an employee's first and last name
- *idNumber*: The idNumber attribute is a string variable that holds an employee's ID number
- *department*: The department attribute holds the name of the employee's department
- *position*: The position attribute holds the name of the employee's job title
- *yearsWorked*: This attribute holds the number of years the employee has worked at the company

The class should have the following overloaded constructors:

- A constructor that accepts the following values as arguments and assigns them to the appropriate data members (attributes): employee's name and employee's ID number, employee's department, employee's position, and the number of years the employee has worked at the company.

- A constructor that accepts the fallowing values as arguments and assigns them to the appropriate data members (attributes): employee's name and employee's ID number. The *department* and *position* attributes should be initialized to the empty string ("") and the *yearsWorked* attribute should be initialized to zero.

- A default constructor (accepts no arguments) that initializes the *name*, *idNumber*, *department*, and *position* attributes to the empty string (""). The *yearsWorked* attribute should be initialized to zero.

Write Get and Set methods for each attribute: *name*, *idNumber*, *department*, *position*, and *yearsWorked*.

- Do not allow the *yearsWorked* attribute to be set to values less than zero. If an attempt is made to set *yearsWorked* to less than zero, do not set the attribute and communicate the problem to the calling program. Do not use cin or cout in the set method.

Remember that the class declaration (.h file) will contain the class attributes and function prototypes. Name the class declaration file *employee.h*. The class implementation file (.cpp) will contain the function definitions (don't forget to include the employee.h file). Name the class implementation file *employee.cpp*.

Next create a program to utilize the Employee class you created in the following manner:

1. CREATE AN ARRAY OF SIZE THREE OF THE DATA TYPE EMPLOYEE. Add the three following Employee objects to your array:

Name	ID Number	Department	Position	Years Worked
Jenny Jacobs	JJ8990	Accounting	President	15
Myron Smith	MS7571	IT	Programmer	5
Chris Raines	CR6873	Manufacturing	Engineer	30

2. Display the data for each of the three employees to the screen (you do not need to display the lines in the table)

The program utilizing the employee class should be in a separate .cpp file. Name this program *employeeTest.cpp*.

KEEP READING – there are two more programs in this assignment, go to the next page.

Homework #7 - Program 2: Stats Class and Rainfall Statistics Program – Based on Chapter 10 Lecture

Design and create a class named *Stats* that has an array of 12 doubles as one of its member variables.

The values in the array should be set by making 12 calls to a public member function named ***setValue*** **that accepts two arguments, an integer indicating which value is being provided (the first number, the second number….etc) and a double holding the actual data value.**

In addition to the *setValue* member function, the class should have the following additional member functions:

- A default constructor that sets all of the values in the array to zero
- A *displayValues* function that displays all of the values in the array (cout is ok for this one)
- A *calcTotal* function that calculates and returns the total of the 12 values in the array
- A *calcAverage* function that calculates and returns the average of the 12 values in the array
- A *calcLargest* function that calculates and returns the largest value in the array
- A *calcSmallest* function that calculates and returns the smallest values in the array

Remember that the class specification/declaration (.h file) will contain the member variables and member function prototypes. Name the class declaration file *Stats.h*. The class implementation file (.cpp) will contain the function definitions (don't forget to include the *Stats.h* file). Name the class implementation file *Stats.cpp*.

Next create a program that uses the *Stats* class to hold rainfall data and report annual rainfall statistics. The program should first create a *Stats* object named *rainfall* and call its *setValue* member function to set each of the 12 monthly rainfall totals to a user entered amount. It should then produce an annual rainfall report that shows the total, average, lowest and highest rainfall amounts for the year. Note that the printing of the rainfall report should be done in the client program file not in the *Stats* class.

Input Validation: If any amount less than 0 is passed into the *setValue* function, a default value of 0 should be used in his place.

The program utilizing the employee class should be in a separate .cpp file. Name this program *raainfall.cpp*.

KEEP READING – there are two more programs in this assignment, go to the next page.

Homework #7 Program 3 – Gift Box Class and Client Program (to give you more practice)

Step 1: Design a *GiftWrap* class

Design a *GiftWrap* class that a gift wrapping store could use to calculate the price of wrapping rectangular gift boxes and generate invoices/receipts for their customers. The class will allow a gift wrapping store to calculate the cost of wrapping various sized rectangular boxes. The class will have member functions to calculate the price of the gift wrap box being wrapped, tax on the gift wrap sold, and the total cost of the gift wrap both including and excluding tax.

The class should have the following member variables:

- ***length***– this variable holds the length of the box in inches
- ***width*** – this variable holds the width of the box in inches
- ***height*** – this variable holds the height of the box in inches
- ***taxRate*** –this variable holds the sales tax rate for the company
- ***pricePerInch***– this variable holds the price per inch of the gift wrap

The class should have the following member functions:

- **Two Constructors:**
  - One constructor should be a no argument (no arg) constructor. This constructor should set the *length, width and height* member variables to 1.0, the *pricePerInch* member variable to 0.0036, and the *taxRate* member variable to 0.08.

  - One constructor that accepts the following *two* values as arguments (so the constructor should have two parameter variables) and assigns them to the appropriate member variables: the tax rate and the price per inch. This constructor should set the *length, width and length* member variables to 1.0.
    - Do not allow the *pricePerInch* member variable to be set to numbers less than zero. Use reasonable default values. It's up to you to know how to design this based on class lecture on object oriented programming.
    - Do not allow the *taxRate* member variable to be set to a number less than zero or greater than one. Use reasonable default values. It's up to you to know how to design this based on class lecture on object oriented programming.

- **Five Set Functions:**
  - Design a set function for each member variable: *length, width, height, pricePerInch,* and *taxRate*.
    - Do not allow the *length, width, height,* and *pricePerInch* member variables to be set to numbers less than zero. It's up to you to know how to design this based on class lecture on object oriented programming.
    - Do not allow the *taxRate* member variable to be set to a number less than zero or greater than one. It's up to you to know how to design this based on class lecture on object oriented programming.

    .
- **Five Get Functions:**
  - Design get function for each member variable: *length, width, height, pricePerInch,* and *taxRate*
  - It's up to you to know how to design these functions based on class lecture on object oriented programming.

- ***calcSubtotal* Member Function:**
  - This member function should calculate the price of the gift wrap BEFORE tax. The price of the gift wrap before tax is calculated as follows:
    1. First calculate the surface area of the box.

- Surface Area = (2 * length * width) + (2 * length * height) + (2 * width * height)

2. Next multiply the surface area of the box by the price per inch. This is the subtotal for the gift wrap.

- ***calcTax* Member Function:**
  - This member function should calculate the tax on the gift. The tax is the subtotal * the tax rate. Be efficient and use existing functions if you can.

- ***calcTotal* Member Function:**
  - This member function should calculate the overall total price of the gift wrap which is the subtotal + the tax. Be efficient and use existing functions if you can.

Step 2: Create a Program to test utilize the *GiftWrap* class

Next create a program to utilize and test the *GiftWrap* class. This program is for a small gift store called "Sally's Gifts". (HINT: keep your program simple)

First steps: First create a string type variable to hold the name of the gift store: "Sally's Gifts". Next, use the constructor that takes two arguments to instantiate a *GiftWrap* object/instance named *sallys*. When you call the constructor pass in .0925 for the tax rate and .0025 for the price per inch of the gift wrap.

Next: Display the following menu:

```
GIFT WRAP INVOICE GENERATOR
-------------------------------
a)Generate Gift Wrap Invoice
q)Quit
```

--If the user enters 'a' or 'A':

- First, ask the user the length of the box being wrapped and call the proper class member function. Then ask the user the width of the box being wrapped and call the proper class member function. Than ask the user the height of the box being wrapped and call the proper class member function. Make sure and handle any input validation issues as needed using the techniques we learned in class during the object oriented lecture (you should know them for listening to lecture).

- Next, display a gift wrap invoice/receipt similar to the once below (make sure and format it) by calling the proper class member functions (use a variable to print the company name, don't hard code it):

```
GIFT WRAP INVOICE - Sally's Gifts
---------------------------------
Box Length: 10.0
Box Width:  10.0
Box Height: 10.0
Price Per Inch: 0.0025

SUBTOTAL:        1.50
TAX:              0.14
            ----------
TOTAL:            1.64
```

--If the user enters anything other than 'a', 'A', 'Q' or 'q':

- Print the message "Invalid Selection"

--The user should be able to generate as many gift wrap invoices for Sally's as they want until they select 'q' or 'Q' to quit

--If the user enters 'q' or 'Q' quit the program