CSCE 121(505-508) **Homework 4**
Due 11:59 p.m., 2/17/14 [150 points]

This Homework assignment is meant to (a) help you understand how the calculator program worked, (b) give you experience writing functions with different arguments, and (c) give you some experience writing basic classes.

All work should be turned into CSNet by the deadline. **In addition, please bring a hardcopy printout of your code to class the next day.** For programs, you need to turn in only the source code (not object or executable code). Your code will be tested using g++: you are welcome to develop in Visual Studio, but please make sure your code also runs in g++.

0) Create a text file, README, in which you:
   a. State the Aggie Honor statement, or else explain why you cannot do so.
   b. List any resources used, outside the textbook and discussions with the Instructor, TA, or Peer Teacher
   c. List any known problems with the assignments you are turning in. For example, if you know your code does not run correctly, state that. This does not need to be a long explanation.
   d. For places where you may have done some additional work, put in a brief summary detailing what you did.

1) [40 points] This part can be written out on paper. For the calculator program, trace through the program to show how the following inputs would be parsed. For each input, you should show:
   - The call stack. That is, show a list of which (see section 8.5.8 for an example) functions are called in which order. Each time a function is called, you should write the list of functions in the call stack (you do not need to show the data stored, except as noted below). You need to list only expression, term, and primary in your list.
   - The status of the token stream class at the time the function is called – what is in the buffer, and whether full is true or not.
   - What is remaining in the input stream at the time the routine is first called.
   - The value returned by each function call (intersperse this in the list of function calls)

   The inputs to process are:
   a. 2.5+3.1; [10 points]
   b. 12*10; [10 points]
   c. (3+2)/5; [15 points]

   Note: you might find it useful to do this using the debugger in Visual Studio. As an example (the real program does NOT work exactly like this!), if you have a program that calls function A, and function A calls functions B and C, and function B calls function D, you could write:

```
A     token stream: buffer: "" full: false     input: 73*7+2;
AB    token stream: buffer: "" full: false     input: *7+2;
ABD   token stream: buffer: "*" full: true     input: 7+2;
D returns 10
AB
B returns 52
A
AC    token stream: buffer: "*" full: false    input: +2;
C returns 9
A
A returns 12
```

   d. [5 points] Answer the following question: Why does this program implement order of operations correctly? That is, why is () handled before *,/, which is handled before +,-?

2) [50 points]Write a program that reads in data regarding employees from a user and then computes some statistical data.
   a. The user should repeatedly input a name (first name and last name), an age (in years) and a salary (as a floating point number). Keep taking input until someone with an age of 0 is entered (and don't count that person). For example, an input might be:

```
John Smith 30 37250.00
Jane Doe 32 42153.99
```

Blah Blah 0 0.0

b. The data should be read in stored in 4 separate vectors, one for first name, one for last name, one for age, and one for salary. [5 points]
c. Write functions for each of the following. The functions should compute the following information, but should not print the information. There should be a single function computing each of these, though that function may call other functions.
    i. The maximum, minimum, and average salary.
    ii. A vector giving the last names in sorted order (alphabetically). ***
    iii. The median age. Hint: you might want to sort the ages, first. ***
d. Allow the user to request any of the following information. Using the functions from part c, compute the relevant data below and print the result to the screen [15 points each]
    i. A list of employees' names, ages, and salaries, in alphabetical order by last name (assume no two people have the same last name).
    ii. The names of people with above average salaries.
    iii. The maximum, minimum, and average salaries of the youngest half of employees (age below the median), and the maximum, minimum, and average salaries of the oldest half of employees (age above the median). Ignore those whose age is the median.

*** Note: sorting is an algorithm. You may either figure out a way to sort the information on your own, or research a method for sorting vectors automatically. In any case, though, you should have a function that returns a vector giving the sorted ages or last names.

3) [60 points] Pretend you are setting up an online store. You should create classes, structures, and helper functions that allow you to implement the following:
a. You want to keep track of products. You will need to keep track of information about the product. Pay attention to what is asked for, below.
b. Keep track of customers. You will need to keep track of what the customer owes and some other information, outlined below.
c. You should create the ability for the user to do the following:
    i. Identify a product that will be for sale. You should read in the name, model number, wholesale cost, and retail cost for the product.
    ii. Enter a new customer. You should read in the customer name and ID.
    iii. Take a shipment of new products. Read in the model number and quantity. If you don't know what the product is that you're getting, reject the shipment, otherwise add that to inventory.
    iv. Let a customer buy something. The customer ID, product model number, and quantity should be taken as input.
        1. If there is sufficient quantity of the product on hand, then the customer should be charged that amount and the product be deducted from inventory.
        2. If there is not sufficient quantity, the sale should be rejected.
    v. Let a customer make a payment. Read in the customer ID and the amount of payment. It's OK for customers to have a positive balance, but they cannot make negative payments.
    vi. Find out about a customer: enter a customer ID number, and print out the customer's name, current balance, *and a list of what the customer has previously purchased*.
    vii. Find out about a product: enter a model number and get the name of the product, the amount that has already been sold, and the amount in inventory.
    viii. Print lists of all information about all customers and all products.
d. Keep in mind that you should structure your classes and functions well and in an organized fashion. Think through what classes you need, what you want to store in each one, and what functions need to be provided. Try to keep information private as much as possible.
e. Grading: Class definitions [15 points]
    Functions [15 points]
    Input/Output [20 points]
    Good design [10 points]