CS 10 - Assignment 6

Collaboration Policy

Collaboration between students on programming assignments is **NOT** allowed under any circumstances - you may not even discuss your work with others; you may **NOT** get coding assistance, or copy code, from **ANY** outside source *(books, web sites, current or past students, your own previous submissions if you are repeating the course, etc.)*. We test every submission for copying and when we find it we treat it as flagrant academic dishonesty on the part of ***all*** parties involved, regardless of their roles as provider or consumer. The penalty is generally instant failure in the course, in addition to the usual sanctions applied by Student Judicial Affairs.

As you can see from the submission header below, we ask you to essentially take an oath that the code you submit is ***entirely your own work*****.**

At the same time, we certainly understand very well that you will frequently need help completing your programming assignment, so we provide channels where you may get that help in a way which will strengthen your programming skills: instructor and TA office hours, and the discussion forums where you can ask all the questions you want about the assignment, and even help others with the understanding that you have gained *(but not, of course, with any actual code)*.

Assignment Submission Instructions

Your assignment must be submitted as a ***simple text file*** named ***main.cpp***
Files in ANY other format (MS Word document, etc.) or with ANY other name (main, main.doc, main.txt, etc.) will ***not*** be graded.

Submit your work via the appropriate iLearn assignment link, making sure you use the correct link and **click on the attach button**

We strongly recommend that:

1. you submit at least *6 hours* before the deadline, even if you haven't completed the program - you can re-submit as often as you like, and we will only grade the final submission
2. once you have submitted your final attempt, go back and download it back to your system: then run it just to make absolutely sure that it really is the file you intended to submit!

You are budding professionals, and are expected to take full responsibility for abiding by the requirements of a contract.

The ***only reason we will ever accept*** for a missed deadline is if the system administrators inform me that either the iLearn system or the CS department servers were off-line for a significant period around the time of the deadline *(In which case we will probably have notified you of alternative procedures)*.

Remember to include the following header information at the top of your program
(DO NOT REMOVE THE // at the beginning of each line, these tell the compiler to ignore these lines)

```
// Course: CS 10 <quarter & year>
//
// First Name:
// Last Name:
// Course username: <enter the username you use to login in the lab>
// Email address: <enter your cs or UCR student email address here>
//
// Lecture Section: <e.g. 001>
// Lab Section: <e.g. 021>
// TA:
//
// Assignment: <assn1, hw2, lab3, etc.>
//
// I hereby certify that the code in this file
// is ENTIRELY my own original work.
//
// =================================================================
```

NOTE: This header MUST appear at the top of EVERY file submitted as part of your assignment
(don't forget to fill in *your* details and remove the <> brackets!!).

Copy & paste this header into your file then update personal details.

Assignment Specifications

You are the manager of an ice cream shoppe. Every week you hold a competition for your cashiers and reward them with a weekly bonus. Normally you just reward the person with the **highest total sales** and the **largest individual sale**. However, starting this week an item of the week is going to be advertised and you would like your cashiers to sell this as much as possible. So, you decide to offer an additional prize of $50 at the end of the week for the cashier that sold the **most of that week's item**. Currently, you have **3 cashiers**: Penny, Leonard and Sheldon.

Since you do not wish to determine the winner by pencil and calculator you write a simple program to help you tabulate the winners based on the transaction logs of the cash registers (which are compiled weekly). You will post the results of your program for all your employees to see, so everyone knows the weekly winners.

The transaction logs are all arranged in the same fashion. The log contains two parts, the first line is the item of the week. The second part is a compilation of transaction summaries. Each contains the transaction number, cashier number and a list of items purchased in that specific transaction. For example a log with a single transaction looks like:

```
10
0001 0 3,12,6,7,10,9
```

As a manager your time is precious, so instead of reproducing code you have written several helper functions already and placed them in an *assn.h* file. **You must utilize these functions when completing the program. Do not** hard code values based on example transactions-- `1213450 0 1,1242,234,5,6` is a perfectly valid transaction.

Your output will all be formatted so it is easy for your employees to determine the winners without needing to read all the information provided. Given the previously mentioned example transaction log, your output would appear like:

```
Item of the Week: 10
Largest Single Sale Seller: Penny
Best Item of the Week Seller: Penny
Best at total sales: Penny

Individual                Penny          Leonard        Sheldon
Total Sales               13.82          0.00           0.00
Largest Single Sale       13.82          0.00           0.00
Item of Week Sales        1              0              0
```

Additional Input and Output

Input File:

```
6
0001 0 3,4,6,7,8,9
0002 0 4,7,8,9
0003 1 3,7,8,6
0004 0 3,4,6,9
0005 2 3,4,6,9
0006 0 7,8,9
0007 1 6,12,6
0008 2 0,1,2,3,4,5,7,8,9,10
```

Output:

```
Item of the Week: 6
Largest Single Sale Seller: Sheldon
Best Item of the Week Seller: Leonard
Best at total sales: Penny

Individual              Penny          Leonard        Sheldon
Total Sales             32.38          16.07          23.27
Largest Single Sale     11.15          8.25           16.62
Item of Week Sales      2              3              1
```

Input File:

```
9
0001 0 3,4,6,7,8,9
0001 1 3,4,6,7,8,9
```

Output:

```
Item of the Week: 9
Largest Single Sale Seller: tie
Best Item of the Week Seller: tie
Best at total sales: tie

Individual              Penny          Leonard        Sheldon
Total Sales             11.15          11.15          0.00
Largest Single Sale     11.15          11.15          0.00
Item of Week Sales      1              1              0
```

Implementation Requirements

You **must** use the functions we have provided in the `assn.h` file (and listed below) when implementing your program. **We will change the values** used in each function when testing your code, so your program will not work if you try to implement this programming assignment without using these functions. For example, the names will be changed, the pricing will be changed, the taxable items will be different, etc. As long as you always use these functions when your program needs to get the price of an item, check if it is taxable, get the cashier's name based on their id number, and determine the winner, your code will work. If you try to do any of these things without using the provided functions, your program will not work when we test it.

We will not change the cashier ID numbers, these are constant: "`0`", "`1`" and "`2`".

You are designing the main function only. Call the provided functions when you need them. **We have provided the `main.cpp`, begin with this!** It has the necessary include statements and a declaration of the TAX constant you should use when calculating the tax on taxable items.

`assn.h` Function Definitions

```
/**
   @brief Acquires the price of a specified item.
   @param number a string value that contains the number of an item
   @return the price before tax of that item
*/
double getItemPrice(string number)


/**
   @brief Determines if an item is taxable.
   @param number a string value that contains the number of an item
   @return true if the item is taxable, false if it is not taxable
*/
bool isTaxed(string number)


/**
   @brief Gets the name of the cashier based on specified ID number.
   @param number a string value that contains the cashier's numeric id
   @return the name of the cashier or tie if there is a tie
*/
string getCashierName(string number)


/**
   @brief Determines the winner ID or a tie if no clear winner exists.
   @param one the amount associated with cashier 1
   @param two the amount associated with cashier 2
   @param three the amount associated with cashier 3
   @return a string containing the winning cashier's numerical ID
           or the word tie if there is a tie between 2 or more of the cashiers
*/
string determineWinner(double one, double two, double three)
```

Development Strategy

You must take this program one step at a time or you will not be able to finish the assignment without getting too much help from others and putting yourself in danger of failing the course and being reported to SCAIP for unauthorized collaboration or cheating.

DON'T move on to the next step until you see the previous step working.

1. Get the item of the week to print out
2. Get a full transaction line, print it out prior to breaking up, to prove to yourself you got it correctly
3. Break up and store each individual "word" in the transaction line and output the value of each of these variables, to make sure you are reading them in correctly.
4. Now break up each individual item number in the 3rd word of your line. Output each item number individually to make sure you are reading them correctly and that you are stopping at and using the last item correctly.
5. Now, instead of outputting each item number, pass it into the function that returns the price of the item and just output this price. Again, verify it outputs the correct prices before moving on.
   (NOTE: We still **have not** read in the next transaction line, still working on the first line.)
6. Next, add tax to the taxable items only (call our provided boolean tax function to determine this).
7. Continually accumulate a total for this one transaction.
8. After accumulation is complete and the one transaction is processed update the variables for the specific cashier that completed this transaction.

STEPS 1 - 8 should work for a single transaction input log.

9. Okay, now you are ready to move on to getting the rest of the transactions by putting the above code into a loop. Just output the value of each transaction for now to make sure you read them all.

Before you turn in REMOVE all excess printouts not displayed in output examples.

Marking Guidelines

- Make sure your code compiles.
- Make sure your output matches the format of the examples: spacing, spelling, etc.
- We utilize the `diff` command to grade so make sure your spelling is correct.

Basic Style Guidelines

- **indentation** - follow proper 4 space indentation as you go into loops or branches
- **spacing** - blank lines should be used to separate logic blocks
- **no line wraps** - no line of code should have more than 80 characters
- "**meaningful**" variable **names**
- **comments** - each logical block of code should have a brief explanatory comment