CSCI 2270 Data Structures and Algorithms Summer 2017
Instructor: Holler/MonteroQuesada
Assignment 1

Due Friday, June 9th, by 8 am Assignment 1 Part A (Pseudocode)
Due Wednesday, June 14th, by 8 am (Implementation)

It's like Craigslist, only different

Objectives

- Review C++ by:
    - Reading a file with unknown size
    - Storing, searching, iterating, and sorting through data in an array of struct

---

Read the entire assignment carefully before beginning.
In this assignment, you're going to develop a simulated community message board that monitors items *wanted* and items *for sale* and looks for matches. When a match is found, e.g. there is a bike for sale for $50 and a bike wanted, where the buyer will pay up to $60, then the item is removed from the message board.

There is a file on Moodle called *messageBoard.txt* that includes up to 100 wanted or for sale items in five categories: bike, microwave, dresser, truck, or chicken. Each line in the file is one item. Your program needs to open the file, read each line, and use an array of *structs* to store the available items. You can assume there will never be more than 100 lines in the file, therefore, you can declare an array of *structs* with a fixed size of 100 to represent the items available on the message board. Each *struct* represents an item and has a type, such as bike or truck, a price, and whether it is for sale or wanted. (You can treat for sale or wanted as an integer or Boolean, where 0 is for sale and 1 is wanted, for example.)

Pseudocode

The first part of this assignment is to turn in pseudocode for a selected portion of the code. For this first assignment, the pseudocode will seem proportionally lengthier and will be based on the significant portion of the assignment: matching the list on file to the array. This will include tasks such as reading in the messageBoard.txt file, parsing it, and determining the state and contents of the current array (i.e. is there an item match or not).

For more information, look below at the appropriate section describing the above task or ask for help from the instructors, TA, or CAs.

Implementation

Your program needs to read the file until it reaches the end, and you can't assume there will always be 100 lines, there may be less. As lines are read from the file, print each line using the command:

```
cout<<"item read "<<item<<" cost "<<cost<<endl;
```

You also need to check if there is a match with the existing items in the message board. There are two options to consider:

Match is not found in the array

If a match is not found in the array, add the item to the array at the first unused position, e.g. if there are four items, add the item to position five.

Match is found in the array

If a match is found, use the first match found and stop searching the array. Do not add the new item read from the file to the array. Remove the matched item from the array and shift the array to fill the gap left by the removed item. (Section 3.2.4 of your book shows the algorithm for deleting an item from an array and shifting.) Write the action performed to the terminal, formatted as <type><space><price>, such as *bike 50*. Your printing should be done with the command:

```
cout<<itemArray[x].type<<" "<<itemArray[x].price<<endl;
```

where *itemArray* is the array of *structs* and *x* is the index where the item was found in the array. The *type* is one of the following: bike, microwave, dresser, truck, or chicken. The *price* is the actual item cost, not what the user is willing to pay.

Other things your program needs to do

Handle the file name as an input from the user

Require the user to enter the name of the file to open. You will be running your code on the CodeRunner autograder, and the filenames we use will not be *messageBoard.txt*. You can use *cin* or *getline*, such as

```
cin>>filename;
```

or

```
getline(cin, filename);
```

Print sorted array contents after all lines read from file

After all lines have been read from the file and all possible matches have been made, there will be items left in the array that no one wanted. Include a function in your program that prints out the final state of the sorted message board (by item name increasing order) along with the total number of leftover items and call the function after displaying the matched items.

The function parameters and return values are at your discretion, but the function needs to correctly print the contents of the sorted array using the command:

```
cout<<itemArray[x].type<<","<<"for sale"<<","<<itemArray[x].price<<endl;
```

for “for sale” items

```
cout<<itemArray[x].type<<", "<<"wanted"<<", "<<itemArray[x].price<<endl;
```

for “wanted” items and

```
cout<<arraySize<<endl;
```

for total # of leftover items.

Count loop iterations

As part of your program, you need to count the number of times that loops execute. Iteration of a loop is an example of an operation that scales with the size of the input data that needs to be processed. The block of code that reads each line from the file depends on a loop because the number of lines in the file can change if you use a different input file. The code needed to search the message board, or shift data in the array, also scale with data input because the size of the message board will change as items are added and removed.

At the end of your program, after displaying the items left on the message board, calculate the total number of loop iterations needed to read the file, search the array, and shift the array when an item is deleted. Each count should be formatted as:

```
file read:6
search array:8
shift array:2
```

And submitted to Part C of the assignment.

Loop iterations example

After reading the first line in the file, your *fileCounter* variable should be 1 for reading one line from the file.

When the first sale occurs, all loop iterations needed to complete the sale should include: 1 for the read from the file, *x* for the number of iterations it takes to find the matching item in the array, and *y* for each shift in the contents of the array.

Format and ordering of program output

You will be submitting your assignment to the codeRunner, so it's important that the output of your program is formatted and ordered in a certain way. You should use the *cout* statements given in the above sections and output your results in a certain order. The sample output is as follows:

```
item read xbox cost 60
item read ps4 cost 100
item read xbox cost 50
item read xbox cost 60
item read ps4 cost 110
ps4 100
#
xbox
xbox
xbox
3
#
```

Loop iterations

Use the # as the delimiter for each output type, the coderunner will look for that character. Don't output anything other than what's specified, it will confuse coderunner. Even the seemingly harmless newline character could spell doom.

Submitting Your Assignment:

Submit the pseudocode for matching the list to the array using Assignment 1 Part A link.
Submit your code to CodeRunner (on Moodle) using the Assignment 1 Part B link :
Submit your count on Moodle using the Assignment 1 Part C link :

CodeRunner will run its tests and display the results in the window below the submit/check box. If your code doesn't run correctly on CodeRunner, read the error messages carefully, correct the mistakes in your code, and check again. You can modify your code and check it as many times as you need to, up until the assignment due date. However, a good practice is to write your code on your IDE and test it there. Afterwards, you can submit the working code on the CodeRunner submit box.
Make sure your code is commented enough to describe what it is doing. This might not be useful for CodeRunner, but it will if you are chosen for the interview.

Include a comment block at the top of the .cpp source code with your name, assignment number, and course instructor.
If you do not get your assignment to run on CodeRunner, you will have the option of scheduling an interview grade with your TA to get a grade for the assignment. Even if you do get the assignment to run on CodeRunner, you can schedule the interview if you just want to talk about the assignment and get feedback on your implementation.

What to do if you have questions?

There are several ways to get help on assignments in 2270, and depending on your question, some sources are better than others. There is a discussion forum on Piazza that is a good place to post technical questions, such as how to shift an array. When you answer other students' questions on Piazza, please do not post entire assignment solutions. The CAs are also a good source of technical information, especially questions about C++. If, after reading the assignment write-up, you need clarification on what you're being asked to do in the assignment, the TA and the Instructors are better sources of information than the CAs. We will be monitoring Piazza regularly.