CSCI 2275 Programming and Data Structures

Fall 2020

Instructor: Hoenigman

Assignment 2

Due: Wednesday, Sept 9 (Monday lecture) and Friday, Sept 11 (Wednesday lecture)

**Community Garage Sale**

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 Canvas called Assignment2.zip that includes the class outline for your code and a text file called *garageSale.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 complete the *MessageBoard* class to provide two types of functionality: first, process the file by reading each line and matching items for sale with items wanted. Once all file items are processed, the user should be able to enter individual items for sale and wanted and either identify matches for those items, or add them to the message board.

Use a vector of *structs* to store the available items. Each *struct* represents an item and has a type, such as bicycle 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.) Your program needs to read the file, and as lines are read from the file, you 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 vector**

If a match is not found in the vector, add the item to the end of the vector, e.g. if there are four items, add the item to position five.

**Match is found in the vector**

If a match is found, use the first match found and stop searching the vector. Do not add the new item read from the file to the vector. Remove the matched item from the vector. Write the action performed to the terminal, formatted as <type><space><price>, such as *bike 50*. Your printing can be done with the command:

cout<<itemVector[x].type<<” “<<itemVector[x].price<<endl;

where *itemVector* is the vector of *structs* and *x* is the index where the item was found in the vector. 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 a command line argument**

Require the user to enter the name of the file to open as an argument to the program. When we test your code, the filename we use will not be *garageSale.txt*. We will run your program from the command line, such as

./Assignment2 testFile.txt

where *Assignment2* is the name of the executable they build when they compile your code, and *testFile.txt* in the filename. Your program needs to use *argv[1]* as the filename in the ifstream.

**Print 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 vector that no one wanted. Include a function in your program that prints out the final state of the message board, 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 vector using the command:

cout<<itemVector[x].type<<”, ”<<”for sale”<<”, “<<itemVector[x].price<<endl;

for “for sale” items and

cout<<itemVector[x].type<<”, ”<<”wanted”<<”, “<<itemVector[x].price<<endl;

for “wanted” items.

**Format and ordering of program output**

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 the following order:

Items sold.

#

Items remaining in the message board after reading all lines in the file.

#

Don’t output anything other than what’s specified.

**User Input**

After the file is processed and the remaining items in the message board are displayed, your program should display a menu with options to post an item for sale, post an item wanted, print the message board contents, and quit the program. The code provided on Canvas has that menu already included and the basic structure for handling user input. When the user enters posts and item for sale or wanted, your code should check the existing message board for a match, print the same message you did when matching items in the file, and remove the item from the message board.

**How to know if your output is correct?**

There is a Slack channel for this class and you are welcome to post the output you get for the garageSale.txt file on Slack and ask if other students get the same answer. If you want to test your code with a smaller data set, create a new .txt file and use that file as the input file when you run your code. Please don’t post your code on Slack.

**Building Your Code:**

To build the code, you need to use the terminal and the command:

g++ -std=c++11 assignment2.cpp MessageBoard.cpp -o messageBoard

You can then run the program from terminal using:

./messageBoard

**Submitting Your Code:**

**Submit your assignment to Canvas**

Zip your Assignment2.cpp, MessageBoard.cpp, and MessageBoard.h and submit them as Assignment2\_<LastName>.zip to Canvas using the Assignment 2 Submit link. Make sure your code is commented enough to describe what it is doing. Include a comment block at the top of each .cpp file with your name, assignment number, and course instructor.

**What to do if you have questions**

There are several ways to get help on assignments in 2275, and depending on your question, some sources are better than others. We’ll use Slack as our discussion forum, and that is a good place to post technical questions, such as how to shift an array. When you answer other students’ questions on Slack, please do not post entire assignment solutions. Your TA is 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 Instructor is the best source of information.