CSCI/CMPE 1370

Homework Assignment #4: Managing structured data

Submitting:
Once again, you will have the option to work in randomly assigned pairs. The ability to work effectively with others is a critical skill. The teamwork policy is described in the course syllabus. You will turn in **one** copy of the assignment as a team. Your submission must have the following comment at the top:

```
//*****************************************
// First and last name of team member 1, First and Last Name of team member 2
// CSCI/CMPE 1370
// Homework Assignment #4
//*****************************************
```

Part of this assignment is to come up with a set of tests that prove that your program works. **Include sample runs** of the program. These test runs can be copied into the source file as a block comment at the bottom of your code file, or copied into a separate text file and submitted along with your code file.

As before, there is a self-evaluation form to help you evaluate your work and show that you are able to do so. Copy the form below into a new text file, fill it out, and submit it with your code. The numbers in parentheses are the max points for each requirement.

*When you are satisfied that your solution is complete, submit it **through blackboard.** Your submission should include at least one .cpp file and one .h file containing your program, a separate .txt file with your test output, and a separate .txt file with your self-evaluation.*

The problem:
Write a C++ program to read in inventory information, sort it in different ways, and display it. Put your class definition and functions in a separate header file.

Part 1: Read in the data and store it

This program deals with inventory records for a set of items. For each item there is an id string, the quantity sold and the quantity remaining. This data is stored in the included file (inventory.txt) in the following format. Each record consists of an id string and the two quantities. So for example, the following would be a file with 2 records. The first item, with id `619847GBE`, has sold 641 units and has 998 remaining.

```
619847GBE 641 998
418712IMB 107 867
```

For this assignment, you'll be using the *Object-Oriented Programming* approach, by creating a class to represent inventory items. For part 1, read in and store the data by:

- Define a class called `Item` to hold the data for a single item in *private* data members.
- Write a function (not a class method) to read the inventory information from a data file into an array of `Items`. Your function should be able to read any number of records in from the data file, up to a limit of 1000 (the max array size). **Do not hardcode the number of records to be read in!**
- Declare an array of 1000 `Items` in main and read the inventory data into it.

Throughout this assignment, it's up to you to figure out what constructors and other public methods you need to your class, but leave the data members are private!

Part 2: Make sure it worked

Next, make sure that you can print the items in your array.

- Add a method to your `Item` class that prints itself (a single item). You can print it in the same format that you see in the file, no need for anything fancy.
- Write a function that prints an array of `Item` objects. This is **not** a class method, because it takes an array of objects, not a single object.

As a test, in your main function, after you read in the inventory information, write C++ statements to print the fourth, eighth and sixteenth elements of the array, then print the whole array.

Make your output look like:

```
Fourth record: 981836KEA 747 171
Eighth record: 706756EUU 687 730
Sixteenth record: 359989USA 934 972

Initial array:
619847GBE 641 998
418712IMB 107 867
227451GEM 789 181
981836KEA 747 171
986516IGU 303 71
501024BMU 895 743
455559SKK 764 234
706756EUU 687 730
500635MIS 841 670
341955IUS 32 907
791357GEK 391 284
933292AMK 581 901
929002IKA 20 177
474556IBK 532 537
342188MKI 649 367
359989USA 934 972
529875KIB 653 655
706784GGS 596 345
764956SEB 839 304
105170KKU 613 130
```

Part 3: Sort the array

Now you are going to write two functions (not class methods) that each take an array of `Item` objects and sort the array by different criteria.

- The first function should sort the array by the number of units sold (greatest to least)
- The second function should sort the array by the number of units remaining (least to greatest)

Included below is an implementation of the *selection sort* algorithm for sorting an array of integers. You can copy and modify this function to work with `Item` objects. Take your time to read through it and understand what it is doing! I recommend drawing memory with example numbers to help you see how the algorithm works.

As before, add additional public methods to your `Item` class as necessary.

```
void selectionSort( int list[], int length )
{
	//local vars
	int smallestPos;
```

```
        int temp;

        // for every position in the array
        for( int i=0; i<length; i++ )
        {
              // find the smallest element starting at that position
              smallestPos = i;
              for( int j=i+1; j<length; j++ )
              {
                    if ( list[j] < list[smallestPos] )
                    {
                          // found a smaller one, remember it and keep going
                          smallestPos = j;
                    }
              }

              // see if we found something smaller than list[i]
              if( list[i] > list[smallestPos] )
              {
                    // we did find a smaller one, so swap with list[i]
                    temp = list[i];
                    list[i] = list[smallestPos];
                    list[smallestPos] = temp;
              }
        }

        // done sorting the array
}
```

In main, simply call each sorting function and then display the sorted array. The correct, complete output is included below. **Pay close attention to whether your output is actually correct!**

```
Fourth record: 981836KEA 747 171
Eighth record: 706756EUU 687 730
Sixteenth record: 359989USA 934 972

Initial array:
619847GBE 641 998
418712IMB 107 867
227451GEM 789 181
981836KEA 747 171
986516IGU 303 71
501024BMU 895 743
455559SKK 764 234
706756EUU 687 730
500635MIS 841 670
341955IUS 32 907
791357GEK 391 284
933292AMK 581 901
929002IKA 20 177
474556IBK 532 537
342188MKI 649 367
359989USA 934 972
529875KIB 653 655
706784GGS 596 345
764956SEB 839 304
105170KKU 613 130

Sorted by units sold (greatest to least):
359989USA 934 972
501024BMU 895 743
500635MIS 841 670
764956SEB 839 304
227451GEM 789 181
455559SKK 764 234
981836KEA 747 171
```

```
706756EUU 687 730
529875KIB 653 655
342188MKI 649 367
619847GBE 641 998
105170KKU 613 130
706784GGS 596 345
933292AMK 581 901
474556IBK 532 537
791357GEK 391 284
986516IGU 303 71
418712IMB 107 867
341955IUS 32 907
929002IKA 20 177

Sorted by units remaining (least to greatest):
986516IGU 303  71
105170KKU 613 130
981836KEA 747 171
929002IKA  20 177
227451GEM 789 181
455559SKK 764 234
791357GEK 391 284
764956SEB 839 304
706784GGS 596 345
342188MKI 649 367
474556IBK 532 537
529875KIB 653 655
500635MIS 841 670
706756EUU 687 730
501024BMU 895 743
418712IMB 107 867
933292AMK 581 901
341955IUS  32 907
359989USA 934 972
619847GBE 641 998
```

Self-Evaluation Form

item class private data members (25):
item class constructors (25):
item class public print method (25):
item class other public methods (50):

read function (50):
print array function (50):
first sort function (50):
second sort function (50):

main function printing test items (25):
main function printing arrays (50):

Good coding practices:
Style (indenting & spacing, reasonable identifiers, file structure) (25):
Compiles and runs (25):
Sample test runs (25):
Self-eval (25):

Comments: