COMP 215 Algorithms

Lab #5

This work should, if possible, be completed during the lab and submitted electronically before 11:59pm Friday. I kept this one short since you have an assignment to hand in.

In this lab, you will practice input and output from a file, with a small twist. The input file will have a bizarre format, with lots of data that has to be ignored. Clearly that file was conceived by a complete idiot.

First, we create a data structure to hold the data from the file

1. Write a class `inventoryItem` that contains three *private* data members: an int `isbn`, a float `price` and a string `title`
2. Set this class to be its own friend, so that the methods in the class can access the private data members of other objects of the same class
3. Write a public constructor that takes an int, a float and a string, and initializes the data members in the obvious way. *Do not* write a default constructor.
4. Write a public function `toString` that takes no input, and returns a string of the form:

   ```
   Title:  [put title here] ISBN: [put isbn here] Price:  $[put price here]
   ```

5. Overload the == operator in two ways:
   (a) The first takes an inventoryItem as input, and returns true if the two inventoryItems have the same isbn.
   (b) The second takes an int as input, and returns true if the inventoryItem you are in has an isbn equal to the input.

All the other functions should be in the general scope.

6. Download the file"InputData.zip" from onCourse. Extract the file "barn.moo" (and other examples of input files) from this .zip file. This will be your initial input file.
   The first line from the file barn.moo contains an integer. This integer is the remaining number of lines in the file. Do not hardcode that integer in your program, I will be testing your program with files of variable length.
   The following lines of the file have the following format:

   oink oink [integer] woof [float] quack roar roar [string] meow

   Take a look at the file to see exactly what it looks like.

7. Write a function `readFile` that takes a string containing a file name as input and returns an *array of pointers* to inventoryItem. The function should open the file whose name is given as input, read it and put the data in the array. The function should also return the size of the array by reference.

8. Write a function `printFile` that takes an array of pointers to inventoryItems, an integer containing the size of the array, and a string containing a file name as input. The first line of the output file should contain the number of remaining lines in the file, and each following line should be written using the `toString` function of each object.

9. Write a function `searchArray` that takes an array of pointers to inventoryItems, an integer containing the size of the array, and an integer containing an isbn. The function should return a pointer to an inventoryItem whose isbn is equal to the input, or a NULL pointer if no such object is found.

Finally, write a `main` function that can be used to test all the functions above. The function should use `readFile`, allow the user to search for a number of ISBNs, and then use `printFile`.