CSC 260.002 Programming Assignment #7
Due Date: Tuesday, October 25

This program involves sorting and searching a 1-D array of Strings and using disk files. You will write your program using methods. The program will input a list of Strings from a disk file (use hasNext to determine when you reach the end of the file) into an array, copy the array into a second array, sort this second array in ascending order using Bubble Sort, input a second list of Strings from a second file storing them into a third array, and iterate through that third array to take each String of that array and search for it in both the unsorted and sorted arrays using the sequential search and binary search respectively. You will implement the two searches in separate methods. Each search method will return the total number of comparisons made until it found (or did not find) the target String (rather than returning its location). The output of the program will be the average number of searches required for both searches so you can compare their performances. The program will require 1 input method (you will reuse this method for the two input files), a sort method, two search methods and main. The program will work as follows:

- in main declare 3 String arrays but do not instantiate any of them. Here, we are calling them list1, list2 and targets.
- main will call the input method as in `list1 = getInput(filename);` where `filename` is a String of the name of the file containing the input. NOTE: you can hardcode this name or use a separate Scanner to input it from the user
  - the input method will declare an array of 100 Strings and fill the array with inputs. Use `while(yourscanner.hasNext())` to control the loop, but also make sure you do not exceed 100 inputs in your while loop condition. You will need a counter to count each input (which will also be used to insert the next String into the array). After inputting the file, close the Scanner. Now, in this method, create a second String array whose size is equal to the number of input Strings. Copy the input String array into this second array and return this second array. The reason we are doing this is to ensure that the array size is equal to the number of items in the array so that we can later control access to the array using array.length.
- main will call the sort method passing it the unsorted list as in `list2 = sort(list1);`
  - in the sort method, create a temporary local array, copy the array passed into the method into this temp array, and sort and return the temp array. In this way, list1 will remain unsorted while the returned temp array is referenced by list2 so that you have both an unsorted and a sorted array. To sort, use the parameter's length to determine the size of the temp array and the number of loop iterations in the Bubble Sort. Sort in ascending order. Remember, these are Strings, do not use > to compare two Strings.
- main will call the same getInput method above, passing it the second filename which will input the new file and return an array of the Strings it found. Assign this returned array to your third array as in `target = input2(filename2);`
- in main, using a for loop, iterate through each String in the target array, taking that String and passing it and the appropriate array to the appropriate search method. Remember these searches will return the number of comparisons. Add these to running totals. For instance, you might have the following:

```
unsortedCount += sequentialSearch(list1, target[i]);
sortedCount += binarySearch(list2, target[i]);
```

  NOTE: not all search items will be found in the arrays.
- finally, main will compute and output the average number of searches as doubles, in a formatted way, to demonstrate the difference in search efficiency.

Run your program on the two sets of inputs on the website. The expected output for set 1 is 10.56 for sequential search and 4.89 for binary search. Hand in your commented code and the output of the second set of inputs.