SYSC 2100, Winter 2013
Assignment 4: Searching in a 2-Dimensional Array
Due: March 20, 2013
(Due at noon that day)

We discussed a number of search strategies in class, for example sequential search and binary search. All these search strategies worked on a one-dimensional data structure (array, list). Assume that we have a 2-dimensional data structure stored in an array, as shown below. The data in each row and each column is sorted in increasing order.

1	4	55	88
7	15	61	91
14	89	90	99

A brute-force search solution would iterate over all elements in the array to search for a specific item. If, in general, the array has `MAXROW` rows and `MAXCOL` columns, such a brute-force approach would result in an execution time that is O(`MAXROW * MAXCOL`). If these two values are the same and equal to `n`, the runtime complexity of this brute-force approach is $O(n^2)$.

Devise a more efficient search algorithm for this type of 2-dimensional sorted array. Describe, in a separate document (Word or PDF) the search strategy and argue why your solution is asymptotically more efficient. For a square array, the runtime complexity of your solution should not exceed O(n). Hint: think about binary search along the main diagonal and how that can help to reduce (in a recursive fashion) the problem by eliminating areas of the matrix that will not include the explored element for sure.

The course website provides a Java program `Driver.java` that creates such a 2D sorted array, and invokes specific searches on them. Complete the submission by implementing class `SearchArray` that provides two distinct search methods:

```
/** Finds an object in a 2D array where data
       in each row and in each column is sorted
       in increasing order.
       @param data  a 2D array of Comparable objects
       @param desiredItem  what we are searching for
       @return true if the desired item is found in the array */

public static <T extends Comparable<? super T>>
          boolean search(T[][] data, T desiredItem)

/** A brute-force, inefficient search used merely to check
       the correctness of our more efficient search. */

public static <T extends Comparable<? super T>>
          boolean bruteForceSearch(T[][] data, T desiredItem)
```

`bruteForceSearch`() implements the brute-force approach outlined above, iterating over all array elements with runtime complexity $O(n^2)$. `search` () implements your more efficient strategy, with runtime complexity O(n). Note that your solution should not depend on the matrix being square, MAXROW and MAXCOL are independent positive integers.

Submission Requirements: Submit your assignment using WebCT. Your program should compile and run as is in the default lab environment, and the code should be well documented. Submit all Java class files without using any archive or compression as separate files. You need to submit at least the following two files:

- `SearchArray.java`: provides your implementation of the two search routines
- `Readme.docx` (or `Readme.pdf`): your description of the search strategy and the discussion of its runtime complexity.

If your solution relies on additional class files you implemented, include those in your submission as well.

Marks will be based on:

- Completeness of your submission
- Correct solution to the problem
- Following good coding style
- Sufficient and high-quality in-line comments
- Adhering to the submission requirements

The due date is based on the time of the WebCT server and will be strictly enforced. If you are concerned about missing the deadline, here is a tip: multiple submissions are allowed. So you can always submit a (partial) solution early, and resubmit an improved solution later. This way, you will reduce the risk of running late, for whatever reason (slow computers/networks, unsynchronized clocks, failure of the Internet connection at home, etc.).

In WebCT6, you can manage the submission until the deadline, taking it back, deleting/adding files, etc, and resubmitting it. The system also provides online feedback whether you submitted something for an assignment. It may take a while to learn the submission process, so I would encourage you to experiment with it early and contact the TA(s) in case you have problems, as only assignments properly and timely submitted using WebCT will be marked and will earn you assignment credits.

Bonus part (probably quite challenging, but still no extra marks): you can use these files to write a program to play a generalized version of Six Degrees of Kevin Bacon (http://en.wikipedia.org/wiki/Six_Degrees_of_Kevin_Bacon). Given the names of two actors/actresses, what is the shortest link to connect them via co-stars? For example, the connection between Avril Lavigne and Kevin Bacon is as follows:

```
Avril Lavigne co-starred with Jackie Burroughs
      in "Going the Distance" in 2004
Jackie Burroughs co-starred with Kevin Bacon
      in "Cavedweller" in 2004
```

Another example, starting with Vivien Leigh:

```
Vivien Leigh co-starred with Jill St. John
      in "The Roman Spring of Mrs. Stone" in 1961
Jill St. John co-starred with David Alan Grier
      in "The Player" in 1992
David Alan Grier co-starred with Kevin Bacon
      in "The Woodsman" in 2004
```

Here is one high-level idea to solve this problem: Finding the shortest connection can be mapped to a routing problem in networks. You model the actors as the nodes, and the movies they co-star in as link between those actors. Then you apply a well-known shortest-path routing algorithm such as Dijkstra's algorithm with link costs being one (http://en.wikipedia.org/wiki/Dijkstra's_algorithm), and you got your result. There are a number of online sites that allow you to verify your results and that display the information in a number of ways, such as Find the Bacon (http://findthebacon.com/).

Submission requirements: Submit your solution using WebCT. Your program should compile and run as is in the default lab environment, and the code should be well documented. Marks will be based on:

- Completeness of your submission
- Correct solution to the problem
- Following good coding style
- Sufficient and high-quality in-line comments
- Adhering to the submission requirements

The due date is based on the time of the WebCT server and will be strictly enforced. If you are concerned about missing the deadline, here is a tip: multiple submissions are allowed. So you can always submit a (partial) solution early, and resubmit an improved solution later. This way, you will reduce the risk of running late, for whatever reason (slow computers/networks, unsynchronized clocks, failure of home Internet connection, etc.).

In WebCT, you can manage the submission until the deadline, taking it back, deleting/adding files, etc, and resubmit it. The system also provides online feedback whether you submitted something for an assignment. It may take a while to learn the submission process, so I would encourage you to experiment with it early and contact the TA in case you have problems, as only assignments properly and timely submitted using WebCT will be marked.

Submission Requirements: Submit your assignment (the source files) using WebCT. Your program should compile and run as is in the default lab environment, and the code should be well documented. Submit all Java class files without using any archive or compression as separate files. The main program should be called `TestSet.java`, the implementation of your `LinkedSet` should be in file `LinkedSet.java`. Marks will be based on:

- Completeness of your submission

- Correct solution to the problem
- Following good coding style
- Sufficient and high-quality in-line comments
- Adhering to the submission requirements

The due date is based on the time of the WebCT server and will be strictly enforced. If you are concerned about missing the deadline, here is a tip: multiple submissions are allowed. So you can always submit a (partial) solution early, and resubmit an improved solution later. This way, you will reduce the risk of running late, for whatever reason (slow computers/networks, unsynchronized clocks, failure of the Internet connection at home, etc.).
In WebCT6, you can manage the submission until the deadline, taking it back, deleting/adding files, etc, and resubmitting it. The system also provides online feedback whether you submitted something for an assignment. It may take a while to learn the submission process, so I would encourage you to experiment with it early and contact the TA(s) in case you have problems, as only assignments properly and timely submitted using WebCT will be marked and will earn you assignment credits.