CSC 355. Discrete Structures & Basic Algorithms

Project 1

General Guidelines.

The instructions given below describe the required methods in each class. You may need to write additional methods or classes that will not be directly tested but may be necessary to complete the assignment.

In general, you are not allowed to import any additional classes in your code without explicit permission from your instructor!

Note on academic dishonesty: Please note that it is considered academic dishonesty to read anyone else's solution code, whether it is another student's code, code from a textbook, or something you found online. You MUST do your own work! It is also considered academic dishonesty to share your code with another student. Anyone who is found to have violated this policy will be subject to consequences according to the syllabus and university policy.

Note on grading and provided tests: The provided tests are to help you as you implement the project and (in the case of auto-graded sections) to give you an idea of the number of points you are likely to receive. Please note that the points indicated when you run these tests locally are not your final grade. Your solution will be graded by the TAs after you submit. Please also note that these test cases are not likely to be exhaustive and find every possible error. Part of programming is learning to test and debug your own code, so if something goes wrong, we can help guide you in the debugging process but we will not debug your code for you.

Project Overview.

In this project, you will implement a leaderboard for an imaginary programming competition. For now, the programming competition and leaderboard are simple. Each contestant, who is identified by a name may submit a single solution, which will then be given a numerical score (this is an integer). The leaderboard is meant to maintain a list of the top **M** scores (along with the Contestant's name) that have currently been awarded. No other information needs to be stored at this point.

Important Guidelines:

- Your implementation must work correctly even when you don't know how many scores are coming in.
- The only data structure you are allowed to use is a simple array of size **M**.
- The final leaderboard should be an array of size **M** that contains the top **M** scores (and names) from greatest to least (score-wise).
- Note: Since we haven't covered sorting, you aren't really expected to write a sorting algorithm. However, you are also not allowed to import any sorting methods, which means you either need to implement your own (not encouraged) OR you need to figure out how to solve this without a sorting algorithm. By sorting algorithm, I mean an algorithm that takes a full unsorted array and sorts it.

Required Classes and Methods

`Contestant.java`

Required Method Signature	**Description**
`Contestant(String name, int score)`	Constructor: Creates a new Contestant with the given name and score.
`String getName()`	Return the name.
`int getScore()`	Return the score.
`compareTo(Object o)`	Required for implementing Comparable. Returning a negative number indicates a "less than" relationship. 0 would indicate "equals" and positive would indicate "greater than." You should compare the contestants first by their scores and secondly by their names (alphabetically). The names only need to be compared if the scores are the same.
`toString`	Return a string with the following format: `<name>: <score>`

`LeaderBoard.java`

Required Method Signature	**Description**
`LeaderBoard(int m)`	`Constructor: Creates a new LeaderBoard of size m.`
`void add(Contestant c)`	`Add the contestant to the board` *`if appropriate`*`.`
`Contestant[] finalBoard()`	`Return the final board in sorted order.`

Example.
Input (N=10):
Mary Smith: 60
Anna Adams: 110
Emma Bates: 20
Elizabeth Carson: 106
Minnie Davidson: 65
Margaret Ellis: 64
Ida Frank: 59
Alice George: 92
Bertha Harris: 93
Sarah Ingles: 62
Final Leaderboard (M=5):
Anna Adams: 110
Elizabeth Carson: 106
Bertha Harris: 93
Alice George: 92
Minnie Davidson: 65

Testing

You will need to create a class as well called ***LeaderBoardTest.java*** to test your project.

Submission Procedure.

To submit, upload the following files to D2L and submit them.

Note: Only properly submitted projects are graded! No projects will be accepted by email or in any other way except as described in this handout.

Grading.

Each of the 6 tests are worth 10 points. However, there may be additional deductions depending on your solution, such as

- Not following instructions (up to 100% deduction on that part of the assignment)
- Bad Coding Style (up to 10-point deduction total)
- Implementing something in an unnecessarily inefficient way (up to 25% deduction for that part of the assignment)
- Improper submission (up to 100% deduction on that part of the assignment)
- Does not compile (up to 100% deduction on that part of the assignment)

Regrade Requests

- We do not enjoy giving zeroes, especially when it is because of a small error, so we do allow regrade requests especially in cases where you received a 0 because of a small compiling issue or something else that can be fixed easily. There is still a standard deduction for most of these issues that is generally equivalent to turning the project in three days late.
- All regrade requests should be submitted according to the guidelines in the syllabus and within the allotted time frame.