CS3460: Data Structures

Final Exam

Total Points: 25

Problems

Notes on Grading: Unless otherwise stated, all programs will receive input via `System.in` and will output solutions via `System.out`.

To simplify the grading process, all grading will be automated. When applicable, you will be provided with sample input/output files for testing. You can ensure that your program will receive full marks by testing it with these provided files.

```
$ java YourProgram < input.txt > output.txt
$ diff output.txt correct.txt
```

The first line executes the Java program, redirecting input from a file `input.txt` and writing the output to a file `output.txt`. The second line compares your program's output (now stored in `output.txt`) with the correct answer (stored in `correct.txt`). If these files match exactly, the `diff` program will print nothing. Otherwise, it will list the differences.

Important Note: You are not allowed to use any classes or code from the Java Collections library. While the classes defined in that library would not be an ideal fit for most of our tasks, the purpose of these assignments is to build these data structures from first principles. Programs which import any of these libraries will receive zero points.

Submission: Please submit the files `Island.java` and `Heights.java`, as well as any files needed to compile your program. Please also include your text file for the second half of the second problem.

Honor Code

The following document is an individual exam meant to be worked on and completed by an individual student without the aid of any other student or outside references. Discussing the contents of this exam with another student or exchanging material of any kind with another student constitutes plagiarism. Please review Appalachian State University's guidelines on this matter.

By submitting this exam, you implicitly agree to uphold the honor code. You may use any code posted on AsULearn, any notes you have taken, and any labs you have submitted as reference material, as well as references online for general Java programming help (syntax, semantics, and standard library documentation). You may not search the Internet for specific solutions to problems.

In preparation for this exam, I have found a wide variety of code across the internet that seeks to solve similar types of problems, and I will be using MOSS, a tool for measuring software similarity, to help identify cases of plagiarism.

1. **Eat at Manny's (10 points):** You've come up with an exciting new idea in the world of fast food! People keep getting stranded on desert islands, and you're going to finally take advantage of this hungry untapped market by building fast food restaurants on every island you can find – though it is still unclear how the employees are going to get to work.

   A map is provided to you in a two-dimensional square $n \times n$ grid. Each grid cell either contains the period character `.` (water) or the asterisk character `*` (land). Two land tiles are considered connected if they are adjacent to one another in any of the standard eight directions (horizontal, vertical, or diagonal). An "island" is considered to be all land tiles reachable from one another by a path of connected land tiles. You may consider the map to be surrounded on all sides by water. Please write a program `Islands.java` that reads in a map via standard input and outputs the total number of islands (and therefore, the number of restaurants needed) on the entire map.

   The input contains an integer $n \leq 1000$ on the first line, representing the size of the grid. The following n lines describe the map, with each line containing n characters with dots and stars representing water and land respectively. Your program should only output a single integer representing the number of islands.

`map-1-in.txt`

```
5
...*.
**..*
*....
*.*.*
**...
```

`map-1-out.txt`

```
3
```

2. **New Neighbors (10+5 points):** You run the construction permit office in a small, historic city. As the city has grown, the city has tried to push back on the construction of high-rise buildings. They have come up with a new strategy: the cost of obtaining a permit will be based not on the height of the building being constructed, but on the height of the next-tallest already-built building (the people running the permit office say that this is in the interest of fairness for buildings that were already allowed to be built).

   For instance, assume there are already buildings in town with heights $[5, 2, 17, 9, 3, 6, 10, 14, 11, 7]$. If a client wants to build a building of height $h = 16$, then the cost of the permit will be based on the building with height $h = 14$. Specifically, it is based on the tallest value that is strictly less than the height requested. This matters for when a permit is requested for a building whose height is exactly equal to the height of another building.

   You have compiled a list of all the high-rise buildings in the city. There are n buildings and q permit requests. Each permit includes how tall they plan to build their building. You want to let them know the costs of the permits. For the purpose of this problem, assume that construction is very slow, and therefore the list of buildings is static – a requested permit will not see its construction complete in time to affect the next incoming permit. Please write a program `Heights.java` that accepts a description of all the buildings and incoming permits, and answers by providing a cost for the permits.

   The input begins with two integers $n \leq 10000$ and $q \leq 10000$ on the first line. The following n lines contain the heights of the buildings as integers, one per line. The next q lines are the permit height requests as integers, one per line. The output should contain q lines where each line contains two integers, h and x, where h is the height of the permit requested, and x is the height of the next-tallest already-constructed building, strictly smaller than h. If no such building exists, then $x = 0$ (the permit is free!). The ordering of the output should match the order of the q permits from the input.

heights-1-in.txt

```
10 4
5
2
17
9
3
6
10
14
11
7
3
16
19
1
```

heights-1-out.txt

```
3 2
16 14
19 17
1 0
```

For the remaining 5 points, please submit a text file named `Dynamic.txt` explaining how you would solve this problem if the new construction happened fast enough to affect future permits. That is to say, what if the list of buildings weren't static and when a new permit request came in, you would have to take into account not just the original list of buildings, but all permits that were processed before the current one.

Please explain how your solution would change, if at all, and analyze the runtime of your proposed solution. Please explain it thoroughly enough that one could use your explanation as a blueprint to solve the problem. You may treat any data structures we talked about in class as a "black box" (you do not need to explain how they work, only how you would use them).

Your solution will be graded based on the presence of certain key phrases and an understanding of data structures terminology, as well as an accurate analysis of your proposed solution. Your answer does not need to be long, it only needs to be correct. There are no additional points offered for word count.