04-630: Data Structures and Algorithms for Engineers

Assignment II: Stable Multikey Sorting of Fixation Points in a Scanpath

Deadline: 11:59pm CAT, Friday, 23rd February 2024

Problem Definition

Write a program to produce a sorted list of fixations points, each one specified by its fixation point number and x and y coordinates. The list should be sorted by two keys: the x coordinate and the y coordinate.

The sorted list should have all x coordinates in ascending order. If points have the same x coordinate, the points should be listed so that y coordinates are in ascending order. Points with the same x and y coordinates should be listed in the same order in which they appear in the input, i.e. in ascending order by fixation point number.

Input

The first line of input comprises an integer *N* (1<= *N* <=10); this indicates the number of test cases to be processed, i.e. the number of scanpaths, in the input file. It is followed by *N* sets of fixation points. Each fixation point comprises three numbers: the fixation point number, the *x* coordinate of the fixation point, and the *y* coordinate of the fixation point. The last fixation point in each set is signified by *x* and *y* coordinates equal to minus one, i.e., the (-1,-1) coordinate. You can assume that there are no more than 1000 fixation points in one scanpath sequence and that the coordinates *x* and *y* are in the range 0<= *x, y*<= 2000. Finally, you can assume that all input provided is valid and contains no errors. Therefore, there is no need for input validation.

Output

The output should begin with your Andrew Id. For each test case, write out a sequence of three numbers comprising a fixation point number, the x coordinate, and the y coordinate. Write each set of these three numbers on a separate line. The list should be sorted by x and y, in ascending order. The output should not include the terminating fixation point (i.e. the one with x and y coordinate of minus one). Finally, each list should be terminated by a line of stars/asterisks.

Sample Input

```
2
1 382 359
2 382 358
3 382 357
4 215 242
5 215 241
6 215 240
7 710 245
8 710 245
9 710 245
```

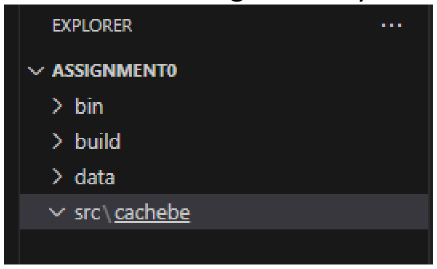
```
10 715 242
11 695 241
12 -1 -1
1 0 0
2 1 1
3 2 2
4 1 1
5 0 0
6 4 4
7 -1 -1
```

Sample Output

```
cachebe
6 215 240
5 215 241
4 215 242
3 382 357
2 382 358
1 382 359
11 695 241
7 710 245
8 710 245
9 710 245
10 715 242
*************
1 0 0
5 0 0
2 1 1
4 1 1
3 2 2
6 4 4
*************
```

Instructions

The input should be read from a file **input.txt** and output should be written to a file **output.txt**. Both files are to be located in the **data** directory. For an assignment, say **assignment0**, you should have the following directory structure. Assume **cachebe** is your Andrew ID:



Submit the following in a **zip file** named with your **Andrew ID** by the deadline shown above.

1. The source code: *.c, *.cpp, and *.h file(s)
2. The cmake file: CMakeLists.txt
3. The test input file: input.txt
4. The test output file: output.txt

Do not include any other files. Submit only the source code files, the CMake file, and the input & output files. Do not include subdirectories.

The source code should contain adequate internal documentation in the form of comments. Internal documentation should include the following.

- Author of the program.
- Functionality of the program.
- Format of input and output to the program.
- Solution strategy: a summary of the algorithm, e.g. using pseudo-code.
- A summary of the way the code was tested, e.g., a description of the different test cases.
- Complexity analysis of the essence of the algorithm in Big O notation with adequate justification.

Place this documentation at the beginning of the **application** file (the file with the **main** function).

Grading

Functionality: 70 Points.

Does the program produce the same output as the instructor test output for the unseen test cases? Zero marks will be assigned for **any test case** that does not produce the required output. There will be fourteen (14) unseen test cases of 5 points each. If the program does not compile and the marker cannot fix the problem in less than five minutes of effort, a total mark of zero will be assigned.

Test Cases: 30 points.

Does the student's test data in the file input.txt contain the following?

- Tests of greater strength than the two examples provided above. [15 Points]
- Tests that check boundary cases. [15 Points]

Internal Documentation.

Marks for functionality and test cases will be awarded if and only if acceptable internal documentation is included. What constitutes acceptability is up to the examiner, but any reasonable attempt will be considered acceptable. Simple one-line descriptions will not be considered acceptable. Therefore, getting zero is plausible.

Use of Standard Template Library (STL)

The use of data structures from STL or similar advanced data structure libraries is prohibited.

Use of Generative AI

The use of generative AI e.g., ChatGPT is prohibited for this assignment. Therefore, any use of generative AI even to generate internal documentation will be considered unauthorized use and will lead to instant penalties as per Academic Integrity Violation (AIV) guidelines.