2018-1 Computer Programming Assignment 4

You should do the assignment on your own. You are not allowed to share codes with others and/or copy codes from other resources. If you get caught, you will lose all points from this assignment.

Grading will be done in Linux environment using Java (OpenJDK) 8, identical to that inside the lab machines. Keep that in mind when writing code in other environments. Programs having any kinds of compile errors will receive neither compile nor test case points.

Do not change the format of input and output. Write everything, including comments, in English.

Upload your work in ETL. You should submit only a single TAR or ZIP file containing those files:
[1] Problem 1: StringSorter.java (S in upper case!).
[2] Problem 2: Maze.java (M in upper case!).

Do not include any subdirectories or any other files inside your archive, so that we can see your source codes right after unzipping it.

Due of this assignment is 11PM on May 30th. No late submission is allowed.

If you have any questions, write an article to the Class Q&A board in ETL so that everyone can see what is going on. TAs will try to respond your questions and announce modifications of the specification promptly, if any. However, TAs will not be able to answer questions after 5PM on May 27th: it is kindly recommended to start your work as early as possible.

Problem 1

Write a Java code StringSorter.java that includes a class `StringSorter`, as follows:

```
public class StringSorter {
	public void selectionSort(String[] arr) {

	}

	public void insertionSort(String[] arr) {

	}

	public void bubbleSort(String[] arr) {

	}

	public void mergeSort(String[] arr) {

	}
}
```

- All methods in the class `StringSorter` get a `String` array `arr` as an input and change its permutation in **lexicographic** order. Refer to Wikipedia articles about how each sort works.
- Only words with lower case alphabets will be used for inputs.
- HINT: Consider each character inside a word as a digit.
- You may add additional methods and/or classes inside StringSorter.java if you need to.
- You may not use `Arrays.sort()` or any pre-implemented sorting methods in Java API.
- Do not include `main()` method inside: graders will test with their own method. For example:

```
public class StringSorterTest {
	public static void main(String[] ar) {
		String[] seq = {"hello", "garbage", "world", "java"};
		StringSorter.selectionSort(seq);
		for(int i = 0; i < seq.length; i++)
			System.out.println(seq[i]);
	}
}
```

If you run the code above, it should print
garbage
hello
java
world\n.

Problem 2

Write a Java code Maze.java that does the following.

- The code reads a maze from input file, given as the first argument (i.e., `ar[0]`).
- A maze is defined as:
  Maze consists of consecutive 1 and 0 in two dimensional space, with width and height being equal.
  We can walk only on 1, we can walk one step up, down, left, right direction.
  We begin from the bottom left-hand corner and maze terminates at the upper right-hand corner.
  Location numbering rule is left-to-right, upper-to-lower. For example, a 3x3 maze could be given as:
  0\t0\t1\n
  0\t0\t1\n
  1\t1\t1\n
  (\t denotes a tab character, \n denotes a newline character.)
  Path of the maze is 7-8-9-6-3.
- Output, which is a file given in the second argument (i.e., `ar[1]`), contains the path of maze, as given above.
- You may assume that width (and height) is not greater than 128.
- Inputs with at least (and at most) one path from the start location to the end location will only be considered. Mazes with cycles will not be considered.
- Possible outputs could be more than one, though you will get full marks **ONLY** when you print the shortest path to exit a maze.
- If you plan to use any data structures mentioned in the lecture (e.g., stack, queue, binary tree, etc.), implement them on your own: do not import any classes directly from API.

<Example>

```
[cp00@cp ~]$ ls
input6.txt      Maze.java
[cp00@cp ~]$ cat input6.txt
0       0       0       0       1       1
0       1       1       1       0       1
0       0       1       0       1       1
0       1       1       0       1       0
0       1       0       0       1       0
1       1       1       1       1       0
[cp00@cp ~]$ javac Maze.java
[cp00@cp ~]$ java Maze input6.txt output6.txt
[cp00@cp ~]$ cat output6.txt /* Shortest path. */
31-32-33-34-35-29-23-17-18-12-6
[cp00@cp ~]$ cat output6-2.txt /* Second possible output. */
31-32-26-20-26-32-33-34-35-29-23-17-18-12-6
[cp00@cp ~]$ cat output6-3.txt /* Third possible output. */
31-32-26-20-21-15-9-8-9-10-9-15-21-20-26-32-33-34-35-29-23-17-18-12-6
[cp00@cp ~]$
```