Lab Test 1 [30 marks]

General Instructions:

1. You will perform the lab test on your personal laptop.
2. You are not allowed to communicate in any way during the test. Disable the following on your laptop before the test begins: wifi, Bluetooth, and any other communication devices (e.g. 3G/4G modems).
3. You can refer to any file on your laptop.

Any violation of the instructions above will result in a zero score for your lab test.

1. Make sure your code can generate exactly the same output as we show in the sample runs. You may get some marks deducted for missing spaces, missing punctuation marks, misspelling, etc. in the output.
2. Do not hardcode. We will use different test cases to test and grade your solutions.

Failure to do the following will attract a penalty of 20% of your score for that question.

1. Include your name as author in the comments of all your submitted source files. For example, if your registered name is "Sotong TAN" and email ID is sotong.tan.2014, include the following block of comments at the beginning of each source file (.java) you write.

```
/*
 * Name: Sotong Tan
 * Email ID: sotong.tan.2014
 */
```

2. Follow standard Java coding conventions (e.g. naming of getter and setter methods, choice of identifier names for classes, methods and variables) as well as indent your code correctly.

3. Ensure that all your Java code can compile without compilation errors. They must compile with any test class(es) that are provided. You may wish to comment out the parts in your code, which cause compilation errors. But remember that commented code will NOT be marked.

Question 1 [10 marks]

1. **[4 marks]** Update `Secretary` according to the class diagram. Run `SecretaryTest` to check your code. Secretary A is considered the same as secretary B if and only if
   i. The `equals` method of `Staff` returns `true` AND
   ii. The boss of A is the boss of B.

2. **[3 marks]** Update `Section.java` according to the class diagram . Run `SectionTest` to check your code. Section A is considered the same as section B if and only if
   i. They belong to the same course,
   ii. Every staff of section A is a staff of the section B, and
   iii. Every staff of section B is a staff of the section A.

   They need not have the same name (e.g. "`G1`" or "`G2`").

1. **[3 marks]** Update the `Sorter` class according to the class diagram.
   a. It implements the `java.util.Comparator` interface and compares two `Staff` objects based on a list of sorting criteria specified by its attribute criteria: `criteria`.

   b. Each `Criterion` object has 2 attributes.
      i. Attribute `field`'s possible values are the constants `NAME`, `SCHOOL`, `YEAR` and `CLASS`. For the field `CLASS`, it will sort based on the class in the following order: `Faculty`, `TA` and `Secretary`.
      ii. Attribute `isAscending`. If true, sort the possible values in ascending order. Otherwise, sort the possible values in descending order.
      iii. For example, if field's value is `NAME`, compare two staff by their name
         1. If `isAscending` attribute is `true`, compare the names in alphabetical order (case sensitive); i.e. "`abs`" is before "`abt`".
         2. If `isAscending` attribute is `false`, compare the names in reverse alphabetical order (case sensitive); i.e. "`abt`" is before "`abs`".

   c. If there are N criteria, compares the two `Staff` objects by the 1st sorting criterion, 2nd criterion, …, then the Nth criterion.

   d. Implement the constructor, `sort()` and `compare()` methods. Run `SorterTest.java` to check your code.

Question 2 [10 marks]

Gomoku, also called Five in a Row, is a board game in which two players take turns to place a stone (either 'X' or 'O') on a 15 x 15 board. The winner is the first player to form an unbroken chain of five stones horizontally, vertically, or diagonally. This is similar to Tic-Tac-Toe where the winner is the first player to form an unbroken chain of 3 stones (instead of 5) on a 3 x 3 board (instead of 15 x 15).

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14
00	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
01	#	#	0	#	#	#	#	#	#	#	#	#	#	#	#
02	#	#	#	X	#	#	#	#	#	#	#	#	#	#	#
03	#	#	#	#	O	#	#	#	#	#	#	#	#	#	#
04	#	0	0	0	O	O	X	#	#	#	#	#	#	#	#
05	#	#	#	O	X	X	O	O	#	#	#	#	#	#	#
06	#	#	#	X	O	O	X	X	#	#	#	#	#	#	#
07	#	#	#	#	X	O	X	X	#	#	#	#	#	#	#
08	#	#	#	#	#	X	O	#	X	#	#	#	#	#	#
09	#	#	#	#	X	#	#	O	#	O	#	#	#	#	#
10	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
11	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
12	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
13	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
14	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#

Legend:
`'X'`: Player A, `'O'`: Player B, `'#'`: Empty Cell

Tasks:

1. Implement `GameException`. It is an **checked** exception.
2. Implement `readFromFile` method in `Utility` class.
3. Implement the constructor, `count`, `placeMove` methods in `Gomoku` class.

Note:

a. Specific instructions are written in the source files.
b. These classes contain test code to help you determine if you have written your methods correctly. Read and understand how each test class works so that you can use it to your advantage.

Question 3 [6 marks]

Consider the class diagram given in the appendix, and complete the following:

1. Organize the source files in the source folder according to the class diagram so that your `compile.bat` and `run.bat` will work correctly.

2. Update the Java source files to include the correct package and import statements with the help of the class diagram.

3. `VendingMachineTest` contains the `main()` method to test the above classes.
   a. Write a one-liner `compile.bat` such that classes are compiled to folder `production`.
   b. Write a one-liner `run.bat` to run the `main` method in `VendingMachineTest`.

4. The directory should look like this:

```
Q3\
 |- config
     |- log4j2.properties
 |
 |- target
 |   |- rolling  // log4j creates this directory
 |       |- rollingtest.log
 |
 |- production
 |   |- <your generated classes here>
 |
 |- source\
 |   |- ...
 |
 |- libraries
 |   |- prebuilt\
 |       |- special.jar
 |
 |   |- log4j2
 |       |- log4j-core-2.11.2.jar
 |       |- log4j-api-2.11.2.jar
 |
 |- compile.bat
 |- run.bat
```

1. If `run.bat` runs successfully, the target folder will be populated with a `rollingtest.log` file after you run `VendingMachineTest`. The content of the file will look similar to

```
2019-03-27 15:56:07,495 DEBUG v.e.VendingMachine [main] Creating an instance of
Vending machine
```

The console output is as follows:

```
numCash = 10
numCash = 110
Result: Pass
```

Question 4 [4 marks]

1. Implement the `reverse` method in `LinkedListUtility.java`. Given a singly linked list

    head -> L_0 -> L_1 -> … -> L_{n-1} -> L_n

    Reverse the nodes in the list so that the

    head -> L_n -> L_{n-1} -> … -> L_1 -> L_0

    Your solution **MUST** have a time complexity of O(n), and space complexity of O(1).

2. Implement the `interleave` method in `LinkedListUtility.java`. Given a singly linked list

    head -> L_0 -> L_1 -> … -> L_{n-1} -> L_n

    Re-arrange the nodes in the list so that the

    head -> L_o -> L_n -> L_1 -> L_{n-1} -> L_2 -> L_{n-2}

    Your solution **MUST** have a time complexity of O(n), and space complexity of O(1).