Data Structures and Algorithms

[Student Name] *[Matr. Number]*
[Email] *[Lab Group]*

Assignment 1 Report

Submitted Documents

- `ArrayUtility.java` – `ArrayUtility` class
- `testArrayUtility.java` – class that tests `ArrayUtility`

  Instructions: Method `testMethodsLab1` is the final method that calls all test methods. This is the only method called in the `main` method of `testArrayUtility.java`. So, the tests for all methods can be called by running the class `testArrayUtility.java`.
- `runTimeTestMaxSort.java` – class that generates running times for two versions of `Maxsort`

  Instructions: Method `compareMaxSortAlgs` calls all methods that generate running times for the two versions of `Maxsort`, so it is enough run this method to obtain running times. This method is the only method invoked in the `main` method of `runTimeTestMaxSort.java`. So, the measurements of running times can be obtained by running the class `runTimeTestMaxSort.java`.

Running Time Experiments

I used the following sizes for random arrays:

$$10, 10, 100, 100, 1000, 1000, 10000, 10000, 100000, 100000$$

I repeated each size to lower the possibility for outliers. The following are the running times in nanoseconds I obtained for `MaxSort` using `shiftRight` method:

11730	3171
142330	179961
3692436	2215182
41814677	42257959
3559183002	3458047857

These are the running times in nanoseconds I obtained for `MaxSort` using the `Swap` method:

6654	1395
14350	21122
398978	382964
24520133	24016434
2425156654	2338061780

The experiments show that `MaxSort` using `Swap` gives better running times—after the sizes of 10000 using `Swap` is approximately 2 times faster.
This is expected as the number of operations using `Swap` is lower than using `shiftRight`.

Acknowledgment of Collaborators. *[Acknowledged students]*