**Sorting Project Name: Shreyash Ranjan**

Adapt your ArrayPlus file to add in methods for selectionSort and insertionSort. (Given below)

Then, without changing the sorting algorithms, modify the 3 sorting methods to count how many times the ArrayList is accessed (calls to get(), set(), remove() or add() methods)

Create a replicate() method that returns a new ArrayPlus object whose ArrayList contains the same elements as the original. You will need 3 versions of the ArrayPlus object to compare the efficiency of the three algorithms for each of the following situations. In each case, rank the 3 algorithms from lowest to highest number of accesses. Enter the results along with the number of times they access the list in the Google Doc provided. Make each ArrayList have 100 values, from 1 to 100.

1. *Sorting a completely randomized list*
2. *Sorting a presorted list*
3. *Sorting a list that is already sorted in reverse order (highest to lowest)*
4. *Sorting a constant list. (all entries are the same value)*

Submit your code for both files along with this document including your results.

NOTE: You will need to create a reverse sorting algorithm for number 3. This can be done by creating a reversed version of any one of the three algorithms used.

RESULTS (rank the algorithms from best to worst in each situation)

1. Completely Random

| Algorithm | Number of times ArrayList was accessed |
| --- | --- |
| bubbleSort | 18612 |
| selectionSort | 5534 |
| insertionSort | 4554 |

1. PreSorted

| Algorithm | Number of times ArrayList was accessed |
| --- | --- |
| bubbleSort | 9900 |
| selectionSort | 5150 |
| insertionSort | 198 |

1. Reverse Sorted

| Algorithm | Number of times ArrayList was accessed |
| --- | --- |
| bubbleSort | 19808 |
| selectionSort | 5522 |
| insertionSort | 5152 |

1. Constant Array

| Algorithm | Number of times ArrayList was accessed |
| --- | --- |
| bubbleSort | 79800 |
| selectionSort | 5150 |
| insertionSort | 198 |