Team Contributions:

Brandon Blank:

- Set up the main function to test algorithms.
- Helped instantiate and implement the template class.
- Wrote insertionsort algorithm.

Fadi Bitar:

- Helped instantiate and implement the template class.
- Wrote bash script to run all commands necessary for running times spreadsheet.
- Wrote quicksort algorithm.

Joshua Bolan:

- Set up the main function to test algorithms.
- Wrote selectionsort algorithm.

Questions:

1. The best performed algorithm for data <= 1000000 is quicksort since the sorting times do not change much when the number of data to sort increases. Quicksort may be the fastest for unsorted data, but is not the fastest for sorted or constant data. If we were to take the averages of all three algorithms, the quicksort algorithm ends up having the smallest average running time.
2. The best performed algorithm for 1000000 < data < 1000000000 is also quicksort since it was the only algorithm that was able to run for all types of data and return a result on time. Selection sort was the slowest in this data size range as it did not retrieve any results when it ran in data size greater than 1000000.
3. Yes, we were able to run most of the algorithms in that data range. Before we changed the stack size, we weren't able to run anything past data size 500000 since stack size was too small.
4.

	Best-case complexity	Average-case complexity	Worst-case complexity
InsertionSort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$
MergeSort	$\Theta(n \lg n)$	$\Theta(n \lg n)$	$\Theta(n \lg n)$
QuickSort	$\Omega(n \lg n)$	$\Theta(n \lg n)$	$O(n^2)$

Selection sort: Omega(n^2), Theta(n^2), O(n^2)

Based on the above chart of algorithm complexities and their corresponding sorting times in lecture 3:

- Our Selectionsort for all data types was at or slower than the worst case complexity since we did not receive any times for data size greater than a million.
- Our Insertionsort for all data types was at or slower than their corresponding complexities, yet we did receive all data size sorting times for this sorting algorithm.
- Our Quicksort for all data types was faster than the lecture sorting times table.

Due to the real vs theoretical run time comparison, a pattern was found where if an algorithm took too long to run, it just wouldn't finish sorting. Those that do not finish are at the worst-case complexity, O(n^2), leaving the time to be exponentially correlated to the data size.