1. Who is your programming partner? Which of you submitted the source code of your program?

My partner was Andrew Yang and I submitted the code for our program.

2. How often did you and your programming partner switch roles? Would you have preferred to switch less/more often? Why or why not?

We switched twice while programming, I thought that was a good amount.

3. Evaluate your programming partner. Do you plan to work with this person again?

Andrew was smart and we worked well together I would definitely work with him again.

4. If you had backed the sorted set with a Java List instead of a basic array, summarize the main points in which your implementation would have differed. Do you expect that using a Java List would have more or less efficient and why? (Consider efficiency both in running time and in program development time.)

Using a list would have made the coding a lot easier. The add function would have taken less work because I would not have to extend the array. The clear(), contains(), containsAll(), remove(), iterator(), isEmpty(), size(), toArray() and removeAll() methods would already be implemented. The running time would be slower on some processes like contains and add since the ones provided use linear searches.

5. What do you expect the Big-O behavior of MySortedSet's contains method to be and why?

The big O behavior should be logN() since it uses binary search.

6. Plot the running time of MySortedSet's contains method for sets of sizes 100000 to 2000000 by steps of 100000. Use the timing techniques demonstrated in Lab 1. Be sure to choose a large enough value of timesToLoop to get a reasonable average of running times. Include your plot in your analysis document. Does the growth rate of these running times match the Big-oh behavior you predicted in question 5?

I can't get this question to work, my eclipse can't seem to run add more than around 100,000 times without failing (method runs but never does anything). I was getting around 40 seconds at 100,000.

7. Consider your add method. For an element not already contained in the set, how long does it take to locate the correct position at which to insert the element? Create a plot of running times. Pay close attention to the problem size for which you are collecting running times. Beware that if you simply add N items, the size of the sorted set is always changing. A good strategy is to fill a sorted set with N items and time how long it takes to add one additional item. To do this repeatedly (i.e., timesToLoop), remove the item and add it again, being careful not to include the time required to call remove() in your total. In the worst-case, how much time does it take to locate the position to add an element (give your answer using Big-oh)?

It takes 20 nanoseconds for my method to complete add() once, In the worst case it takes 2logN to locate the position to add an element (logN for contains() to check if the element is already in the set + logN used in the binary search for the position)

8. How many hours did you spend on this assignment?

4-5 hours.