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

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

Sometimes but not too often. I highly prefer being the driver.

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

He is a hard worker but his coding knowledge is a bit lacking. I do however plan on working 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.)

Many of our methods would have been either covered by Java List or made much simpler. Some of the methods may have been more complex to code considering we had less control over the direct implementation. Java list would have made development a lot

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

Contains is based off our binary search algorithm, which has a complexity of log(n), which contains should share.

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?

Unfortunately our code had a couple bugs that prevented us from implementing this.

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)?

Our will be n+log(n), based on the array copying we have to do to accomplish the addition.

8. How many hours did you spend on this assignment? Programming partners are encouraged to collaborate on the answers to these questions

Around 12