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

My partner is Nathan Wilkinson, I submitted the assignment

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

Rather than basing the roles on a time or amount of code we switched when one of us had an idea that was easier to express through code than through English or when one person got tired they would ask the other to take over for a while.

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

Working with Nathan is fantastic, he is a good programmer, has good humor and somehow can work for hours on end with only the occasional break. I defiantly plain to 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.)

I suspect that using an ArrayList or LinkedList would have been less efficient because while adding is faster and doubling the size is faster adding in sorted order would require looping through the list every time you wanted to access anything.

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

I believe it to be O(log(N)) because it calls a method that does 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?

Size	100k	200k	300k	400	500	600	700	800	900	1mil
Time nano	54	63	67	65	69	74	73	76	77	79
Size	1100	1200	1300	1400	1500	1600	1700	1800	1900	2mil
Time	77	80	79	80	83	82	81	84	83	82

Yes the time appears to grow logarithmically.

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)?

10k	25k	50k	70k	100k

5739n	24,867n	53,845n	74,658	101,454

To locate the element it takes O(log(N)) and then to add the element and copy the elements to right of it takes O(N log(N)). The worst case complexity of adding an element to the set is O(N log(N)) because it has to copy every element over.

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

I would estimate 16 hours on the main part of the assignment, another 5 on testing, and another two on timing. All told I would guess we spent about 23 hours.