Quadratic Sort Lab

Part I

1. Create 3 graphs using data from using bubble sort on 1) an array in random order, 2) an array in descending order, and 3) an array in order.
2. Create 3 graphs using data from using insertion sort on 1) an array in random order, 2) an array in descending order, and 3) an array in order.
3. Create 3 graphs using data from using selection sort on 1) an array in random order, 2) an array in descending order, and 3) an array in order.
4. Create a chart and graph for a random array using 3 different sorts.
5. Create a chart and graph for an ordered array using 3 different sorts.
6. Create a chart and graph for an array in descending order for 3 different sorts.
7. Explain why these are called quadratic sorts. O(n2)
8. Which is the most efficient sort of a random array? Why?
9. Which is the least efficient sort of a reverse ordered array? Why?
10. Which of these sort situations will produce a linear relationship O(n). Why?

Part II

In your new groups, write your own quadratic sort. Submit your code. Discuss the algorithm. Compare its efficiency to the 3 quadratic sorts that we studied in class.

```java
import java.util.Random;

public class Sorts {

	private int[] nos;
	private int steps;

	// Constructs a default array of size 10
	public Sorts() {
		nos = new int[10];
		nos[0] = -10001;
		nos[1] = 3;
		nos[2] = 7;
		nos[3] = 19;
		nos[4] = 15;
		nos[5] = 19;
		nos[6] = 7;
		nos[7] = 3;
		nos[8] = 19;
		nos[9] = -100;
```

```java
}

public Sorts(int[] temp) {
	nos = temp;
}

// Constructs an array with size random Sorts from [0,range)
public Sorts(int size, int range) {

}

// Constructs an array of random Sorts [0-range) array of size count with a
// seed
// The seed allows you to use the same set of random numbers

public Sorts(int count, int range, long seed) {

}

// This constructor will create an ordered array of consecutive integers
// true will yield ascending order
// false will yield descending order
public Sorts(int count, boolean ordered) {

}

public int getSteps() {
	return steps;
}

public void display() {
	for (int x : nos)
		System.out.print(x + " ");
	System.out.println();
}

public int[] getNos() {
	return nos;

}

public void swap(int x, int y) {
	int temp = nos[x];
	nos[x] = nos[y];
	nos[y] = temp;
	steps += 3;
}

public void bubbleSort() {
	//Consecutive values are compared and swapped if necessary
	steps = 0;
	boolean swapped = true;
	steps++;
```

```
            int lastSwap = nos.length - 1;
            steps++;
            int temp = 0;
            steps++;
            steps++; // initialize for loop
            for (int i = 0; i < nos.length; i++) {
                  steps += 3; // boundary check, increment,if
                  if (swapped) {
                        swapped = false;
                        steps++;
                        steps++; // initialize for loop
                        for (int j = 0; j < lastSwap; j++) {
                              steps += 3; // boundary check, increment,if
                              if (nos[j] > nos[j + 1]) {
                                    swap(j, j + 1);
                                    swapped = true;
                                    steps++;
                                    temp = j;
                                    steps++;
                              }
                        }
                        lastSwap = temp;
                        steps++;
                  }

            }
      }

      public static void main(String[] args) {

            Sorts one = new Sorts();
            StopWatch timer = new StopWatch();
            timer.start();
            one.bubbleSort();
            timer.stop();
            one.display();
            System.out.println("Default Array Steps:     " + one.getSteps());
            System.out.println("Default Array time: " + timer.getElapsedTime()+ "
milliseconds.");

                        //This is a sample code for testing bubble sort  for data
in  reverse order
            //Sorts two = new Sorts (100000,false);
            // timer.reset();
            // timer.start();
            // two.bubbleSort();
            // timer.stop();
            // two.display();
            // System.out.println("Reverse order Steps: " + two.getSteps());
            // System.out.println("Reverse order time: " + timer.getElapsedTime()+ "
            // milliseconds");
            // System.out.println();
```

```
}
```