CS 284 - Quiz 3

Mehta Nimtrakul
10440530
I pledge my honor that I have abided by the Stevens Honor System.
mnimtrak@stevens.edu

Exercise 1:

1. TRUE

Exercise 2:

1. C

1. E
2. A
3. D
4. B

Exercise 3:

1a. dm (where m is the top of the stack and d is the bottom)

1b. sm (where S is the front of the queue)

2.

```
public Comparable findMax {
        // variable to store max
        Comparable max;
        // temp queue for storing the dequeues during the loop to make sure original queue is intact
        Queue temp = new Queue();
        while (Q.size() > 0){
                Comparable curr = Q.dequeue();

                if (curr.compareTo(max) > 0){
                                max = curr;
                        }
                // store dequeue'd data to temp queue to enqueue later
                temp.enqueue(curr);
        }
        // put elements back in original queue after finding a max
        while(temp.size() > 0){
                Q.enqueue(temp.dequeue());
        }
        return max;
```

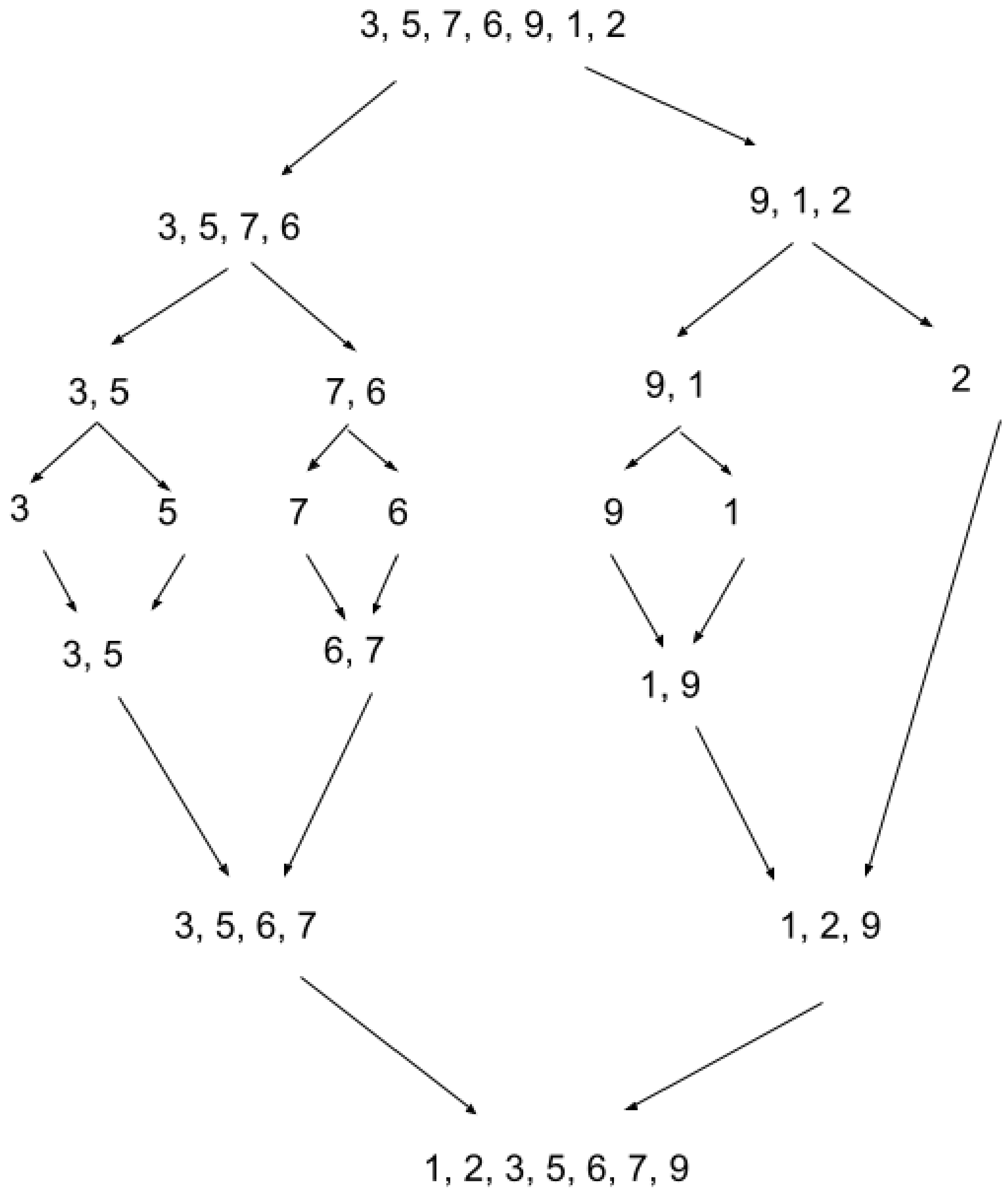
}

Exercise 4:

	Pre-condition	Post-condition	Loop invariant
Insertion Sort	Provided with an unsorted array or list elements that can be compared with a comparison operator.	The array will be sorted in accordance to the comparison operator.	The subarray A[0 to i-1] is always sorted
Selection Sort	Provided with an unsorted array or list elements that can be compared with a comparison operator.	The array will be sorted in accordance to the comparison operator.	1. In outer loop, array is sorted for first i elements. 2. In inner loop, min is always the minimum value in A[i to j]
Bubble Sort	Provided with an unsorted array or list elements that can be compared with a comparison operator.	The array will be sorted in accordance to the comparison operator.	At the end of ith iteration right most i elements are sorted and in place.
Shell Sort	Provided with an unsorted array or list elements that can be compared with a comparison operator.	The array will be sorted in accordance to the comparison operator.	In the main loop it is that the array is gap sorted. The gap is halved on each iteration until it reaches 0. When that happens, the array is sorted.
Merge Sort	Provided with an unsorted array or list elements that can be compared with a comparison operator.	The array will be sorted in accordance to the comparison operator.	At the start of each iteration the nonempty part of the new array contains i - 1 elements of left and right halves(named L and R) of original array in sorted order.L[j] and R[j] are the smallest elements of their arrays that haven't been copied to the new array.

Exercise 5:

```
void mergesort(Comparable[] a, Comparable[] aux, int lo, int hi) {
                    if (hi <= lo) return;
                    int mid = lo + (hi - lo) / 2;
                    sort(a, aux, lo, mid);
                    sort(a, aux, mid+1, hi);
                    merge(a, aux, lo, mid, hi);
          }
```



Exercise 6:

Insertion sort is better for a stream of incoming data. This is because while the stream of data comes in, insertion sort places the new element in linear time. In this same scenario, merge sort would be inefficient since it's not a whole array being split and merged, but a stream of data.