Lab 5

1. Implement an enhanced stack that supports push, pop, peek, isEmpty and also min, so that worst case running time for any operation is still O(1). Write down your idea and your logic for concluding that operations are in every case O(1). Then implement your idea in a Java class called MinStack.

2. Devise an algorithm for reversing the elements in a singly linked list. Implement your solution in code. You can use the singly linked list in the lab folder. What is the running time of your reverse algorithm?

3. Create a sorting routine based on a BST and place it in the sorting environment, distributed earlier. For this exercise, your new class BSTSort should be a subclass of Sorter. Your BSTSort class can be essentially the same as the BST class given in the slides (see the folder in your labs directory for this lab), except that you will need to modify the printTree method so that it outputs values to an array (rather than printing to console).

   After you have implemented, discuss the asymptotic running time of your new sorting algorithm. Run an empirical test in the sorting environment and explain where BSTSort fits in with the other sorting routines (which algorithms is it faster than? which is it slower than?).

4. For each integer *n* = 1, 2, 3,…, 7, determine whether there exists a red-black tree having exactly *n* nodes, with *all of them black*. Fill out the chart below to tabulate the results:

Num nodes *n*	**Does there exist a red-black tree with *n* nodes, all of which are black?**
1	Yes
2	
3	
4	
5	
6	
7	

5. For each integer $n = 1,2,3,\ldots, 7$, determine whether there exists a red-black tree having exactly n nodes, where *exactly one of the nodes is red.* Fill out the chart below to tabulate the results:

Num nodes *n*	**Does there exist a red-black tree with *n* nodes, where exactly *one* of the nodes is red?**
1	No
2	
3	
4	
5	
6	
7	