Final Review

1. Consider a binary tree of height $k$.
   (a) What is the maximum number of nodes?
   (b) What is the maximum number of leaves?
   (c) What is the minimum number of nodes?

2. Consider the following binary tree node class:

   TreeNode.java

   ```java
   public class TreeNode<T> {
   
   private T value;
   private TreeNode<T> left, right; //Children
   //private TreeNode<T> parent; //optional
   
   public TreeNode(T t, TreeNode<T> l, TreeNode<T> r)
   {
       value = t;
       left = l;
       right = r;
   }
   
   public T getValue() { return value; }
   public void setValue(T t) { value = t; }
   public TreeNode<T> getLeft() { return left; }
   public TreeNode<T> getRight() { return right; }
   public void setLeft(TreeNode<T> t) { left = t; }
   public void setRight(TreeNode<T> t) { right = t; }
   }
   ```

   (a) Write a method that gets the size of the tree.

   ```java
   public static <T> int size(TreeNode<T> root)
   ```

   (b) Write a method that gets the number of leaves.

   ```java
   public static <T> int numLeaves(TreeNode<T> root)
   ```

   (c) (⋆) Write a method that prints all paths from the root to the leaves (each path on a separate line)

   ```java
   public static <T> void allPaths(TreeNode<T> root)
   ```
(You will need a helper function.)

(d) (∗) Write a method that determines if the given tree is a BST

```
public static boolean isBST(TreeNode<Integer> root)
```

(You will need a helper function.)

3. Consider the following binary tree:

```
        4
      5   7
   9   12 9 1
```

(a) Give a preorder traversal of the tree.
(b) Give a postorder traversal of the tree.
(c) Give an inorder traversal of the tree.
(d) Give a level-by-level traversal of the tree.
(e) If we stored the above tree the same way we store heaps, what would the resulting storage look like?

4. (∗) Describe how to find the closest (lowest) common ancestor given two nodes in a binary tree (assume each node has a reference to its parent).

5. Using the TreeNode class above, implement a function that prints the nodes out in level-by-level order.

6. Evaluate each of the following postfix expressions, or state that they are invalid.

   (a) 1 2 + 1 3 -
   (b) 1 2 3 4 - - - -
   (c) 2 3 4 + + 5 +
   (d) 2 3 4 * 1 + *

7. Explain how to evaluate a postfix expression.

8. You are given a list of precious stones whose values are given in an array values.

   (a) Suppose the array has values 1,2,3. Draw a tree that shows all possible ways of choosing subsets of the stones, and note the sum of the values in the subset for each subset.

   (b) (∗) Write a recursive function that counts how many subsets of the stones have value that adds up to v. (You will need a helper function.)
public static int numWays(int[] values, int v)

9. Consider the following node class used for storing an arbitrary arity tree (i.e., not just binary).

ListTreeNode.java

    public class ListTreeNode<T> {
        private T value;
        private ListTreeNode<T> firstChild;
        private ListTreeNode<T> nextSibling;
        public ListTreeNode(T v, ListTreeNode<T> f, ListTreeNode<T> n) {
            value = v;
            firstChild = f;
            nextSibling = n;
        }
        public T getValue() { return value; }
        public void setValue(T t) { value = t; }
        public ListTreeNode<T> getFirstChild() { return firstChild; }
        public ListTreeNode<T> getNextSibling() { return nextSibling; }
        public void setFirstChild(ListTreeNode<T> f) { firstChild = f; }
        public void setNextSibling(ListTreeNode<T> n) { nextSibling = n; }
    }

(a) Write a method that gets the size of the tree.

    public static <T> int size(ListTreeNode<T> root)

(b) Write a method that gets the number of leaves.

    public static <T> int numLeaves(ListTreeNode<T> root)

10. Consider the following binary search tree.

```
          10
         /   \
        5    15
     1     12   18
      16    17
```

Perform the following operations in order.

(a) Remove 15.
(b) Remove 10.
(c) Add 13.
(d) Add 8.

11.

(a) Label above (with a 'B') the node that is the successor of 'A'.
(b) Label above (with a 'C') the node that is the successor of 'B'.
(c) What is the balance factor of the root?

12. Give all possible binary search trees for the values 1,2,3.

13. Implement find for a BST.

14. Given a sorted array of distinct integers, explain how to create a BST of minimal height.

15. Start with an empty AVL tree and perform the following operations in order.
   (a) Add 1,3,5,4,6.
   (b) Remove 1.

16. For each, give the worst-case runtime for a tree with \( n \) nodes.
   (a) Find an element in a BST.
   (b) Find an element to an AVL Tree.
   (c) Add an element to an AVL Tree.
   (d) Output the minimum element of an AVL Tree.
   (e) Output the minimum element of a min heap.
   (f) Output the minimum element of a max heap.
   (g) Add an element to a max heap.
   (h) Turn an array into a heap.
17. Consider the following min heap.

(a) What is the index of 11 in the corresponding array?
(b) Add an 8 to the min heap.
(c) Then add a 1 to the min heap.
(d) Then removeMin from the min heap.
(e) Then removeMin from the min heap.

18. Explain in detail how sift-down works on a min heap.

19. Show how to get the 5th smallest element of an array in $\Theta(n)$ time.

20. Consider the array $\{3, 4, 1, 5, 2\}$:

(a) How many inversions are in the array?
(b) Show what steps insertion sort performs on the array.
(c) Show what steps bubble sort performs on the array.
(d) Show what steps merge sort performs on the array.
(e) Partition the array around 3 using the minimum number of swaps possible.

21. Implement merge, which merges two sorted lists.

22. Implement insertion sort.

23. $\star$ Assuming you have access to a function that partitions (as in quick sort), show how to find the $k$th smallest element in expected linear time.

24. Given two sorted arrays $A_1$ and $A_2$, each containing distinct values, compute the number of values they have in common.

```java
public static int numShared(int[] a1, int[] a2)
```

25. For each of the following, give the best and worst-case runtimes of the following:

(a) Bubble sort of an array of size $n$.
(b) Insertion sort of an array of size $n$. 


(c) Quick sort of an array of size \( n \) (using the first element as pivot choice).

26. Consider an array where all of the values are ints between 0 and 100, inclusive. Describe how to determine the number of distinct values in \( \Theta(n) \) time.

27. Name 2 different methods of sorting with worst-case runtime \( \Theta(n \lg n) \).

28. Consider the following graph:

(a) How many connected components are there?
(b) Give the degree of each vertex?
(c) What is the length of the longest path?
(d) How many of the components are trees?
(e) What is the least number of edges that must be removed to yield a forest?
(f) What is the most number of edges that can be added (excluding multiple edges) such that the number of components in the graph doesn’t change?

29. Define a tree.

30. For the graph below, show how it would be stored in an adjacency matrix (for both edges and weights) and in an adjacency list (for both edges and weights).
31. What are the worst-case space requirements for an adjacency matrix and an adjacency list.

32. Give the worst-case runtimes of the following operations for an adjacency matrix and an adjacency list.

   (a) Determine if there is an edge from \( i \) to \( j \).
   (b) Output all of the edges of the graph.
   (c) List the neighbors of a vertex \( v \) (i.e., the vertices \( w \) such that there is an edge from \( v \) to \( w \)).
   (d) List all vertices that have \( v \) as a neighbor (i.e., the vertices \( w \) such that there is an edge from \( w \) to \( v \)).
   (e) Output the degrees of every vertex.

33. (a) What is the worst-case runtime of DFS on an adjacency list?
    (b) What is the worst-case runtime of DFS on an adjacency matrix?
    (c) What is the runtime of Dijkstra’s algorithm?
    (d) What is the runtime of Prim’s algorithm?

34. Consider the following graph.

Below assume lowered numbered vertices are preferred in case of a tie.

(a) Show how Prim’s algorithm finds the minimum spanning tree starting from vertex 2. In case of a tie, prefer lower-numbered vertices.

(b) Compute the distance of getting from vertex 4 to all other vertices using Dijkstra’s algorithm. In case of a tie, prefer lower-numbered vertices.

35. Implement DFS for an adjacency list.

36. Implement BFS for an adjacency list.

37. Explain what algorithm you would use to solve the following problems. Assume the graph is weighted, undirected, and has no negatively weighted edges. In each case give the worst-case runtime. Try to be as efficient as possible.

(a) Compute the shortest path between vertices $v$ and $w$.

(b) Compute the two distinct vertices that are closest.

(c) Compute the two distinct vertices that are furthest apart (they determine the diameter).

(d) Assuming the graph is also connected, compute the minimum cost tree that uses all of the vertices and some of the edges of the graph.

(e) Detect if the graph is acyclic.

(f) Count the number of connected components.

38.
Assume below that in case of a tie, lower numbered vertices are preferred.

(a) Give the order in which DFS visits the vertices.
(b) Give the order in which BFS visits the vertices.

39. Implement the following functions.

(a) Return an array of the degrees of each vertex in an undirected graph:
    \[
    \text{static int}[] \text{ degrees(ArrayList<ArrayList<Integer>> adj)}
    \]

(b) Return the largest vertex index reachable from \( v \) in a directed graph:
    \[
    \text{static int} \text{ largest(ArrayList<ArrayList<Integer>> adj, int v)}
    \]

40. Given an adjacency list for a DAG, give code to find the number of steps in the longest path. You may create an array outside of the function to store needed information.
    \[
    \text{static int} \text{ longest(ArrayList<ArrayList<Integer>> adj)}
    \]