Name: ______________________________ Andrew Id: ______________________________

15-121 Fall 2023 Assessment 4

Up to 50 minutes. No calculators, no notes, no books, no computers. Show your work!

1. (4 points) **Binary Search Tree Construction**

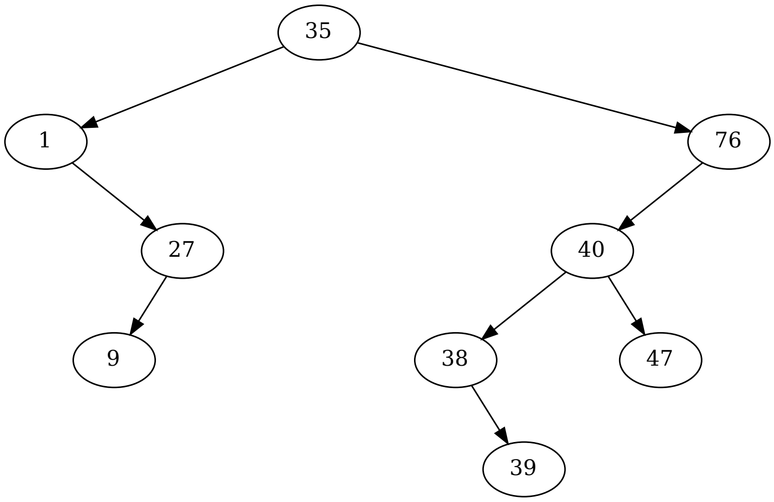
   Imagine you are constructing a binary search tree of integers, and the following integers are added in the following order:

   `73, 82, 68, 59, 46, 72, 5, 33`

   Draw the resulting binary search tree.

2. (4 points) **Binary Search Tree Removal**

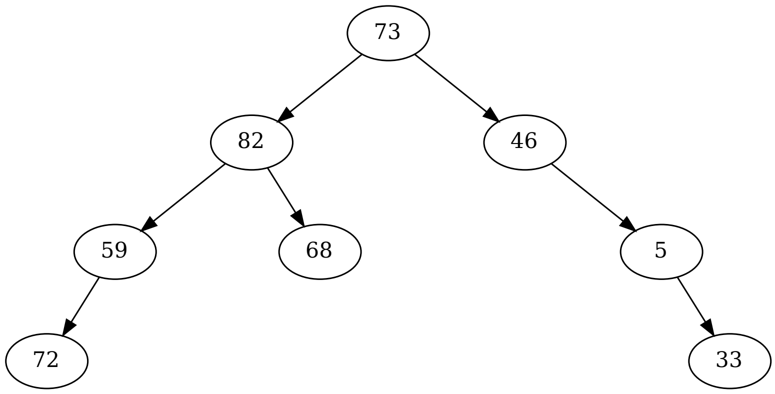
Consider the following binary search tree:



Assuming you are using the in-order successor removal technique discussed in class, draw the state of the tree from after 35 has been removed from it.

3. **Binary Tree Traversal**

Consider the following binary tree. Note that it is *not* a binary search tree, it is simply nodes stored arbitrarily in a binary tree.



(a) (3 points) Assuming the tree is traversed using an *in-order* traversal and the nodes printed, what is the resulting sequence? (Assume that left is followed before right.)

(b) (3 points) Assuming the tree is traversed using a *pre-order* traversal and the nodes printed, what is the resulting sequence? (Assume that left is followed before right.)

(c) (3 points) Assuming the tree is traversed using a *post-order* traversal and the nodes printed, what is the resulting sequence? (Assume that left is followed before right.)

4. **Binary Heap**

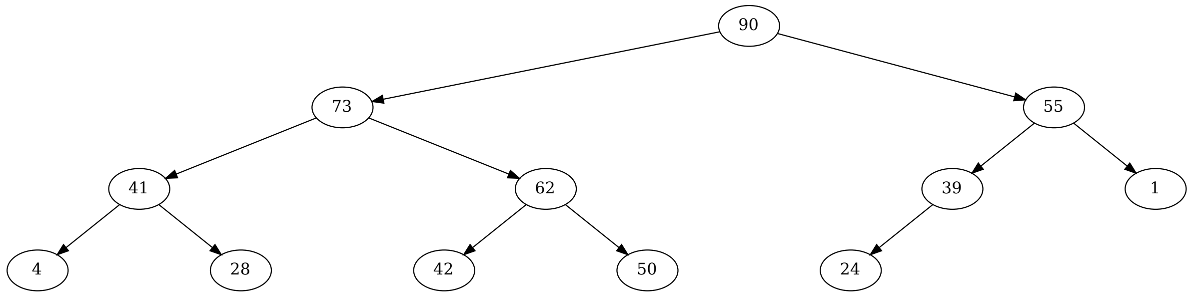
Consider the following array representation of a binary heap:

93	91	65	90	85	38	16	80	3	73				

(a) (1 point) Draw the heap represented by this array.

(b) (3 points) Give the array after adding 92 to the heap. Write your final answer in the boxes below.

5. (a) (1 point) Consider the following heap:



Write this heap in array form. Write your final answer in the boxes below.

(b) (3 points) Building on your answer from part a, give the array after calling `removeMax()` on the heap. Write your final answer in the boxes below.

6. **Binary Search Tree Free Response**

Consider the following code for a binary search tree of strings. The only special thing about it is the helper method called `longer`, which is very useful for solving part (b).

```java
public class BinarySearchTree {
    private TreeNode root;

    private class TreeNode {
        private String data;
        private TreeNode left;
        private TreeNode right;

        private TreeNode(String data) {
            this.data = data;
        }
    }

    public BinarySearchTree() {
        this.root = null;
    }

    public void add(String item) {
        this.root = add(root, item);
    }

    private TreeNode add(TreeNode node, String item) {
        // Assume this code is present and works properly.
        // It uses the natural ordering of strings to
        // place them in the tree.
    }

    private static String longer(String s1, String s2) {
        if (s1.length() > s2.length()) {
            return s1;
        } else {
            return s2;
        }
    }
}
```

The rest of the question is on the next page.

(a) (5 points) Write a new method in this class called `contains(String item)` which returns `true` if `item` is found in the tree, and `false` otherwise. You may write additional helper methods, but you may not modify existing methods or add any new instance variables to `BinarySearchTree`. You also may not use any other data structures, such as an array or an `ArrayList`.

(b) (10 points) Write a new method in this class called `longestString()`, which returns the longest string in the tree. If there are no strings in the tree, return `null`. If there is a tie for the longest string, you may return any one of the correct answers. You may write additional helper methods, but you may not modify existing methods or add any new instance variables to `BinarySearchTree`. You also may not use any other data structures, such as an array or an `ArrayList`.

Hint: Don't forget about the helper method `longer` that has already been provided. It can simplify part of this problem.