Name: _____ Andrew Id: _____

15-121 Fall 2024 Assessment 4

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

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

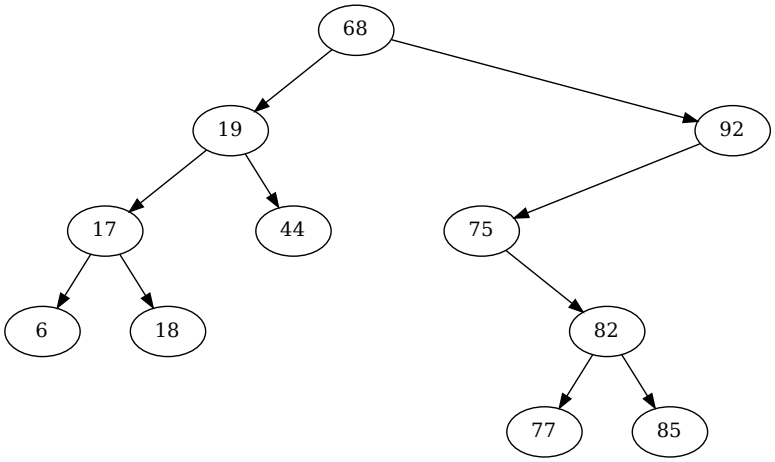
   Imagine you are constructing a binary search tree of integers, and the following integers are added in the following order:

   ```
   21, 23, 78, 80, 50, 77, 54, 79, 7, 99
   ```

   Draw the resulting binary search tree.

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

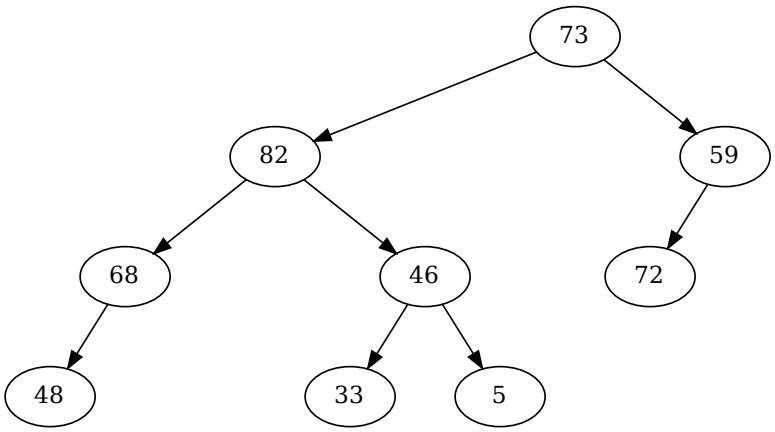
Consider the following binary search tree:



Assuming you are using the in-order successor removal technique discussed in class, draw the state of the tree from after 68 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**

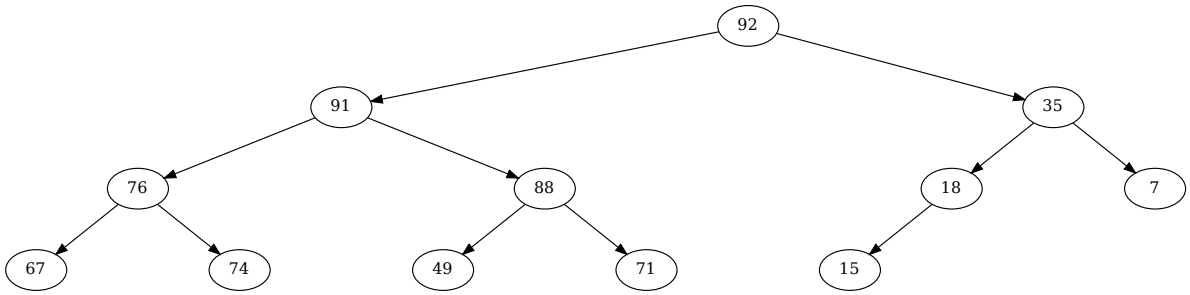
Consider the following array representation of a binary heap:

89	79	72	71	67	27	9	64	62	42				

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

(b) (3 points) Give the array after adding 91 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 integers.

```
public class BinarySearchTree {
    private TreeNode root;

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

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

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

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

    private TreeNode add(TreeNode node, int item) {
        if (node == null) {
            return new TreeNode(item);
        }

        if (item < node.data) {
            node.left = add(node.left, item);
        } else {
            node.right = add(node.right, item);
        }

        return node;
    }
}
```

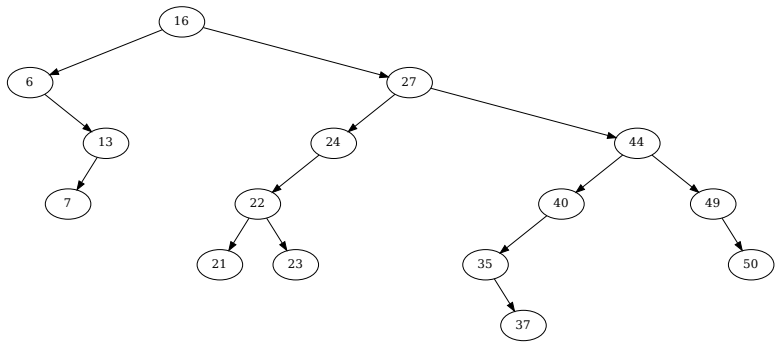
The rest of the question is on the next page.

(a) (7½ points) Write a new method in this class called `valsInRange(int low, int hi)` which returns an `ArrayList` of all of the integers in the BST that are `>= low` and `<= hi`.

For full credit, your solution should be efficient enough to not needlessly search branches of the tree where there can't possibly be any numbers within the range. (For example, if the someone calls `valsInRange(45, 67)` then it would be silly to search nodes located to the left of 41.)

You may write additional helper methods, but you may not modify existing methods or add any new instance variables to `BinarySearchTree`.

(b) (7½ points) Write a new method in this class called `getLevelWidth(int lvl)`, which returns the number of nodes at a specific level in the tree. For example, consider the following BST:



16 is the only node at level 0. 6 and 27 are at level 1. 13, 24, and 44 are at level 2.

The following main function (which builds the BST given above) produces the following output:

```
public static void main(String[] args) {
    BinarySearchTree b = new BinarySearchTree();
    int[] nums = { 16, 27, 44, 40, 35, 6, 13, 49, 24, 50, 37, 7, 22, 21, 23 };
    for (int i : nums) {
        b.add(i);
    }

    for (int i = 0; i < 7; i++) {
        int res = b.getLevelWidth(i);
        System.out.println("Number of nodes at level " + i + " is " + res);
    }
}
```

```
Number of nodes at level 0 is 1
Number of nodes at level 1 is 2
Number of nodes at level 2 is 3
Number of nodes at level 3 is 4
Number of nodes at level 4 is 4
Number of nodes at level 5 is 1
Number of nodes at level 6 is 0
```

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`.

Write your answer on the next page.