

Name: \_\_\_\_\_ Andrew Id: \_\_\_\_\_

### 15-121 Sample Assessment 4

50 minutes. No calculators, no notes, no books, and no computers.

Place answers on the answer sheet.

1. (5 points) **Binary Search Tree Construction**

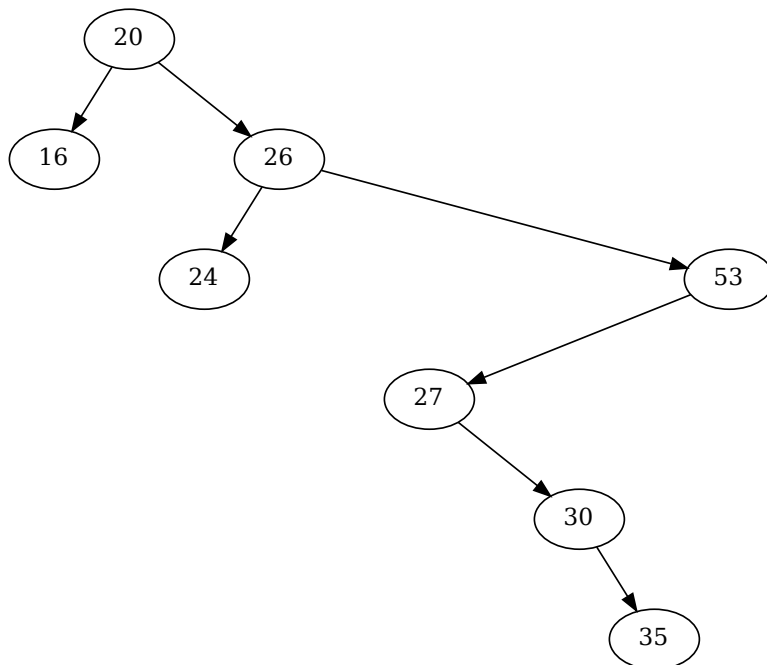
Imagine you are constructing a binary search tree of integers, and the following integers are added in the following order:

16, 67, 85, 55, 11, 43, 71, 87, 3, 44

Draw the resulting binary search tree.

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

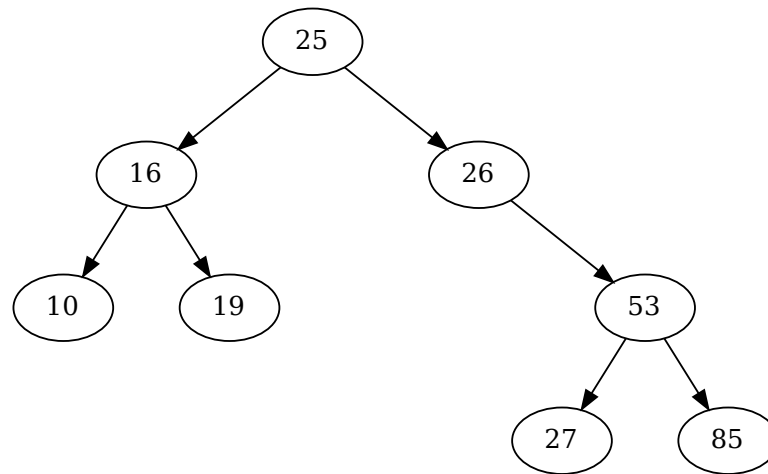
Consider the following binary search tree:



Assuming you are using the in-order successor removal technique discussed in class, draw the state of the tree from after 26 has been removed from it.

### 3. Binary Search Tree Traversal

Consider the following tree:



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

4. (10 points) **Binary Search Tree Free Response**

Consider the following code for a binary search tree of integers. (There is nothing special here, the code is provided just in case you forgot how a binary tree is built.)

```
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;
    }
}
```

Write the code for a new method in this class called `largestEven()`, which returns the largest, positive, even number in the tree. If there are no positive, even numbers in the tree, then return `-1`. 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`.

## 5. Binary Heap

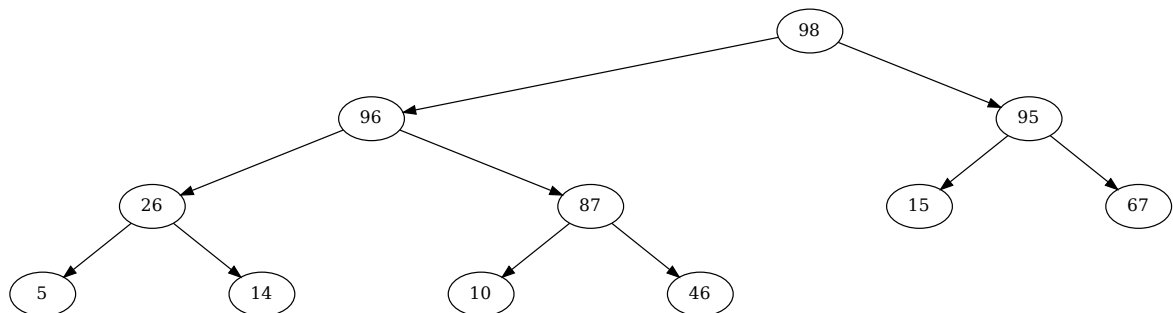
Consider the following array representation of a binary heap:

|    |    |    |    |    |   |   |   |    |   |    |  |  |  |
|----|----|----|----|----|---|---|---|----|---|----|--|--|--|
| 56 | 55 | 39 | 52 | 50 | 3 | 9 | 2 | 49 | 5 | 32 |  |  |  |
|----|----|----|----|----|---|---|---|----|---|----|--|--|--|

- (a) (3 points) Draw the heap represented by this array.
- (b) (3 points) Give the array after adding 45 to the heap.

## 6. More Binary Heaps

- (a) (3 points) Consider the following heap:



Write this heap in array form, assuming that the root is stored in location 0.

- (b) (3 points) Building on your answer from part a, give the array after calling `removeMax()` on the heap.