**NAME: \_\_\_\_\_\_ / 100**

**COMP231 – Data Structures and Problem Solving**

**Spring 2018**

**Midterm Exam #2**

**April 19, 2018**

1. For the binary tree shown below, list the nodes in the order in which they would be visited by:

[4 pts] a. a pre-order traversal

[4 pts] b. an in-order traversal

2. Consider the min-heap shown below. Redraw the heap after each of the following operations are performed. Assume that the heap is reset to the state shown before each operation.

[3 pts] a. add(25)

[3 pts] b. remove()

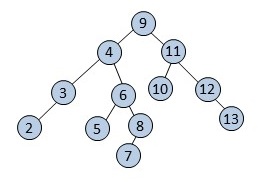
[5 pts] 3. Draw the binary search tree (BST) that would be constructed if the tree were initially empty and then add was called with each of the following elements in order: {50,70,2,458,35,26,0,54,600,601}

4. Consider the BST below: Draw the BST after each of the following operations is performed. You should assume that the BST is reset before each operation.

[3 pts] a. remove(8)

[3 pts] b. remove(70)

5. Consider the following the following AVL Tree:



[4 pts] a. Draw the resulting **BST** after 9 is removed, but ***before*** any rebalancing takes place. Label each node in the resulting tree with its balance factor. At which node is it not AVL balanced and what is the balance factor of that node?

[6 pts] b. Now rebalance the tree that results from the previous step. **Draw a new tree** for each rotation that occurs when rebalancing the AVL Tree. You do not need to label these trees with balance factors.

6. Consider the following binary search tree questions

[5 pts] Suppose the numbers 7, 5, 1, 8, 3, 6, 0, 9, 4, 2 are inserted in that order into an initially empty binary search tree. The binary search tree maintains the Binary Search Tree Property during the insertions. What is the in-order traversal sequence of the resultant tree?

[5 pts] The following numbers are inserted into an empty binary search tree in the given order: 10, 1, 3, 5, 15, 12, 16. Draw the resulting binary tree.

[10 pts] d Recall the BTNode class that we used in our binary tree and Binary Search Tree (BST) implementations.:

**static** **protected** **class** BTNode<K, V> {

**public** K key;

**public** V value;

**public** BTNode<K, V> left;

**public** BTNode<K, V> right;

**public** BTNode<K, V> parent;

**public** BTNode(K key, V value) {

**this**.key = key;

**this**.value = value;

left = **null**;

right = **null**;

parent = **null**;

}

**public** **boolean** isLeaf() {

**return** left == **null** && right == **null**;

}

}

Write a method the prints out the sum of all keys less than 1000. You must use the BST property.

|  |  |
| --- | --- |
|  |  |

[6 pts] 7. Amelia has heard about how great insertion sort is, but is a little concerned about its worst case n2 running time. Therefore, she has decided to make an improved algorithm called ASort. Here is her algorithm: first, she will check if the input matches the worst case for insertion sort. If it does, she will sort it as a special case. Otherwise, she will run insertion sort normally.

1. What is the worst case input for insertion sort? Give an example of size 5.
2. In big-O terms, how long does it take to sort the worst case for insertion sort? Briefly describe the algorithm. (Hint: Don’t use insertion sort.)
3. What would be a worst case input for ASort? How long would it take to sort this input in big-O terms?

[6 pts] 8. Gregory is tired of stable sorting algorithms. Now he wants a completely unstable sorting algorithm. Instead of maintaining the relative order of equal elements, he wants to completely reverse it. Describe how you could modify a sorting algorithm to satisfy Gregory.

5pts :

A binary tree has an inorder traversal of GBHAFJKI and a preorder traversal of ABGHFIKJ. Draw the tree.

10. Back in the days of yore (mid 1990s), contacting a large group of people was very difficult. One solution to this problem was something called a phone tree. Here’s how it would work: Person A would be the central contact for news. When they got a piece of information to pass on, they would call a list of people. Each of those people would have a list for them to call, and so on. Eventually, everyone would have the information. Your job is to design these lists for a few different specifications.

A: (3pts)Suppose every person only calls one other person. What data structure would this call tree resemble?

B. (3pts)Suppose every person calls two other people. What data structure would this call tree resemble?

C. (6pts) Suppose you want to get the news out as fast as possible, i.e. the last person to get the news gets it as soon as possible. Briefly describe how you would structure the call tree. You may assume that every phone call takes the same amount of time.

