Homework 8

ECE 309 Fall 2019
Due: October 23, 2019
Upload an electronic copy of your answers to Moodle under HW8.

*This is a shared google document. This means (1) it may change to clarify content, and (2) other people can view your comments on this file. If you have questions, you are encouraged to comment directly on this document but **do not add your answers here**. Make a copy into your private Google Drive and then edit the document.*

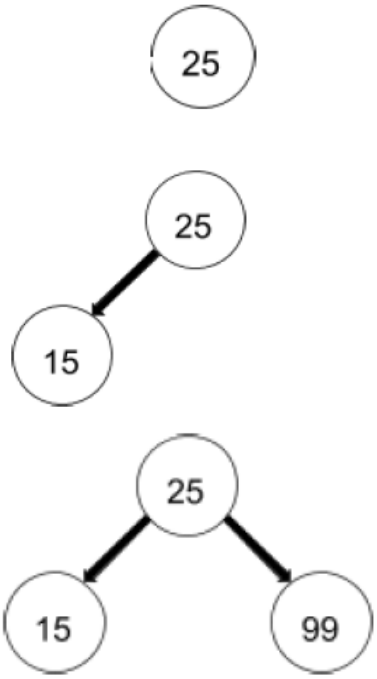
DO NOT ADD ANSWERS TO THE SHARED DOC! THAT'S CONSIDERED CHEATING!

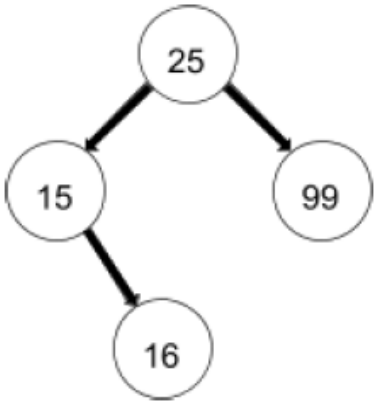
1. Binary Search Tree

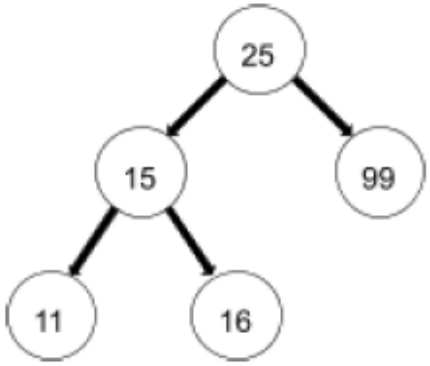
(a) [20 points] Insert the following nodes into a BST: 25, 15, 99, 16, 11, 4, 29, 55. Show the tree after each insertion. Also, determine the following for the final tree:

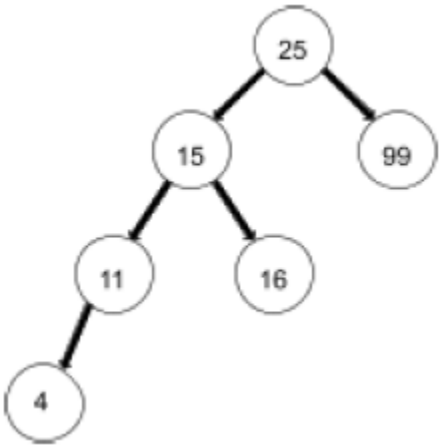
- Depth of each node.
- Height of the tree.
- Number of leaf nodes.
- Number of internal nodes.

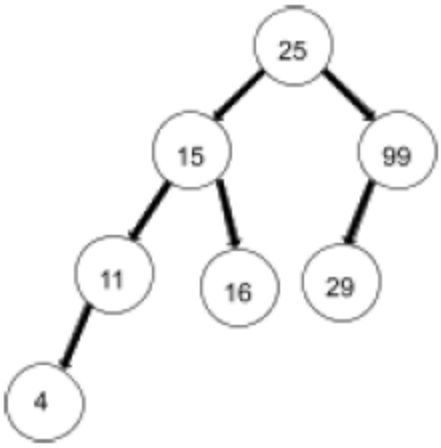
Ans.

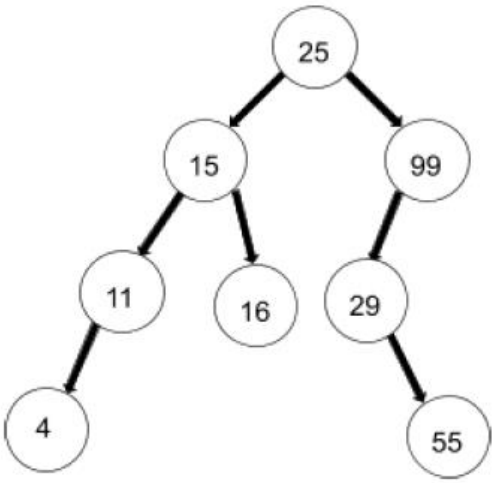



25
15
99
16


25
15
99
11
16

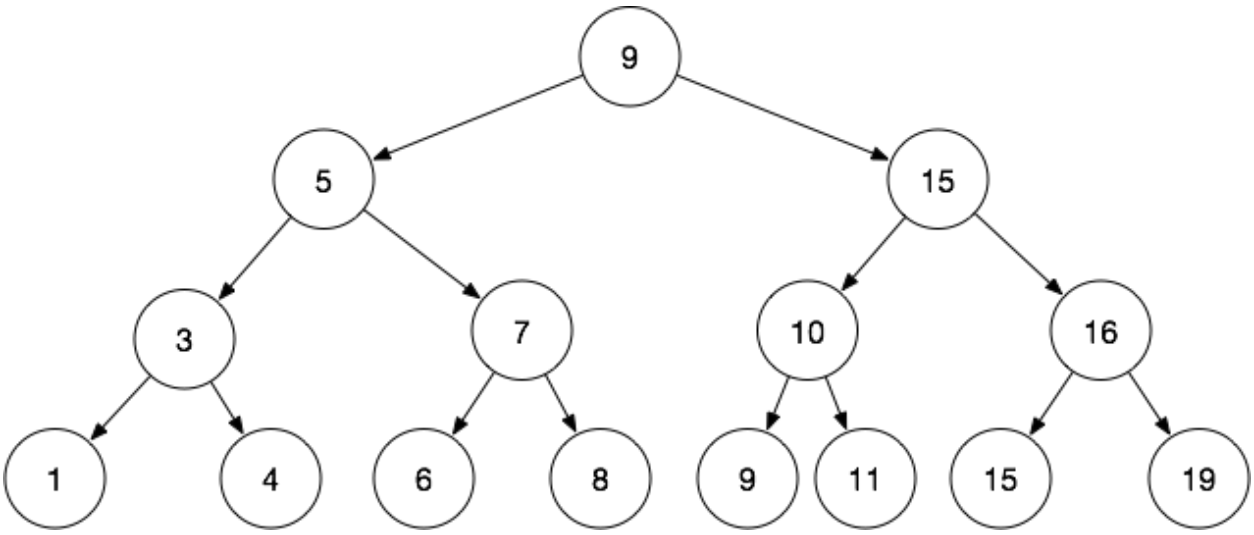

25
15
99
11
16
4





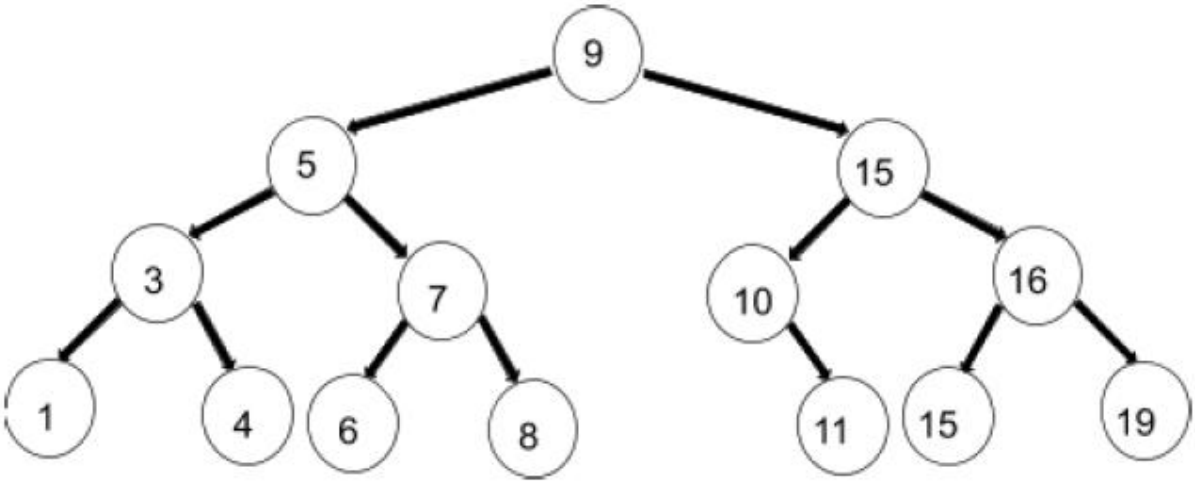
- Depth of each node.
  - Node 25: **depth of 0**
  - Node 15, Node 99: **depth of 1**
  - Node 11, Node 16, Node 29: **depth of 2**
  - Node 4, Node 55: **depth of 3**
- Height of the tree.
  - **height 3**
- Number of leaf nodes.
  - **3 leaf nodes** (Node 4, Node 16, Node 55)
- Number of internal nodes.
  - **5 internal nodes** (Node 25, Node 15, Node 99, Node 11, Node 29)

(b) [20 points] Consider the BST shown below with 9 as the root.

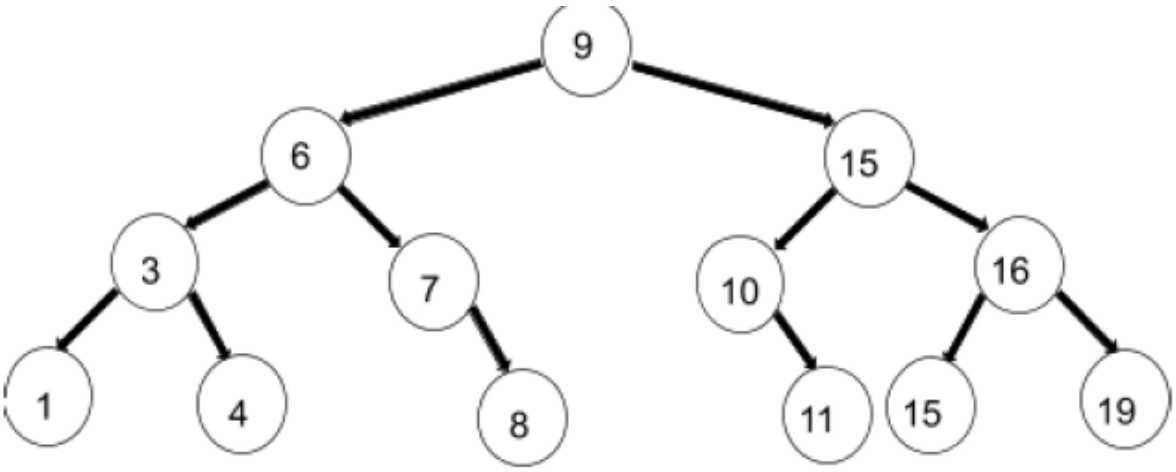


Draw the tree after removing the following nodes: 9, 5, 10, 6. Show the tree after each removal.

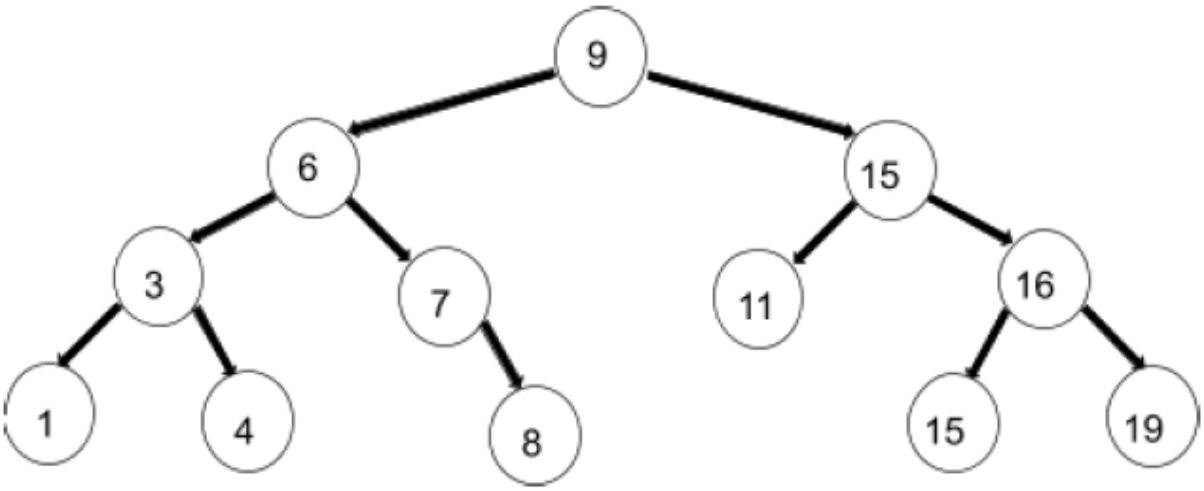
Ans.



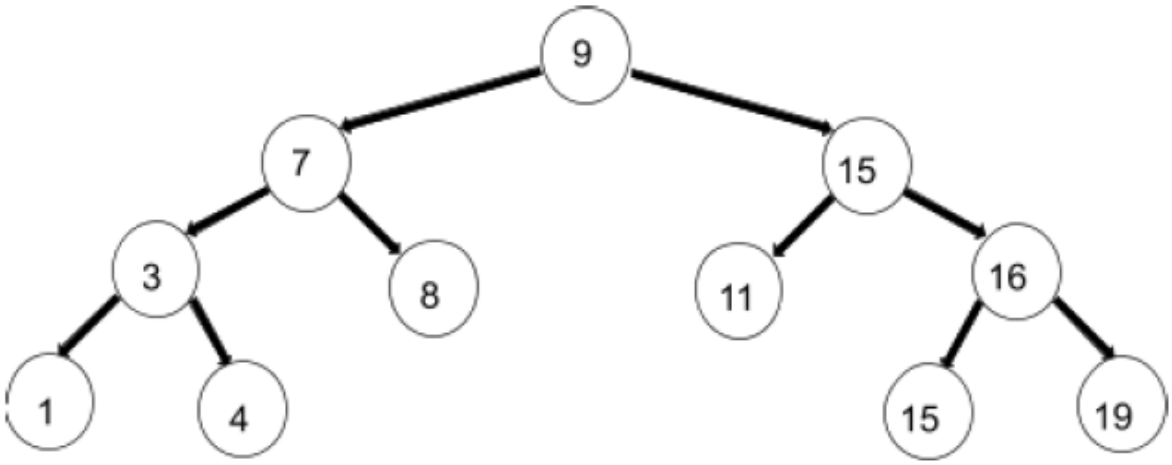
Step 1: Root node 9 removal: Note that its successor is the leaf node 9, so we remove the root node 9 and replace it with the leaf node and above figure shows the final tree after this step.



Step 2: 5’s successor node is 6; left-mode node of the right subtree.



Step 3: successor of 10 is 11; left-most node of the right subtree



Step 4

(c) [5 points] In the BSTRemove algorithm when removing node n, why is it sufficient to only use Case 1 and Case 2 to remove n's successor? Be specific in your answer.

Ans. N's Successor is any node that is larger than all the nodes in N's left sub-tree and smaller than all nodes in N's right sub-tree; This makes searching in the left-subtree of the N redundant; Now, in the right sub-tree, the left most node will be by definition the left-most leaf node or can have upto 1 child (a right-child). Hence, case 1 or case 2 is sufficient; removal of nodes 9, 5 and 6 in the previous problem illustrate this; The property of BST and successor definition enable us in this simplification.

2. ZyLabs

(a) [15 points] ZyLab 16.16. Implement a successor function inside the `BinarySearchTree` class. Given a `BTNode*`, it finds the successor node, if there is one. This is implemented as a private member function since `BTNode` is not visible outside the `BinarySearchTree` class.

To assist with the implementation, I've added a parent pointer to the `BTNode` class and modified the BST implementation to set them properly on insertion. You should use the parent pointer to go up the tree to find an ancestor. For example, this will take you up the tree:

```
parent = node;
while(parent)
   parent = parent->parent;

//after loop, parent is null
```

But, it stops when parent is null. To stop at the root, you might do something like this:

```
parent = node;
while(parent != root)
   parent = parent->parent;

// after loop, parent is root
```

(b) [10 points] ZyLab 16.17. Implement a member function for the BinarySearchTree class that returns an ordered list of all the nodes in the tree. The list must contain the data in order from smallest to largest. The function should run in O(N) time, where N is the number of nodes in the tree. Note, this is essentially an in-order traversal of the tree, but instead of printing out nodes, you place them in a list. I recommend following the same code structure I used in Lecture 15 for in-order traversal in which you implement a helper function to perform the recursive traversal for you.

(c) [30 points] ZyLab 16.18. Implement node removal functionality for the BinarySearchTree. The base remove function has already been implemented for you,

but you need to fill in the implementation for Case 1, Case 2, and Case 3 of the algorithm as described in the lecture notes. The pseudo-code in the lecture should work with minimal modifications. However, it would be a good learning exercise to implement them from scratch if you have time.