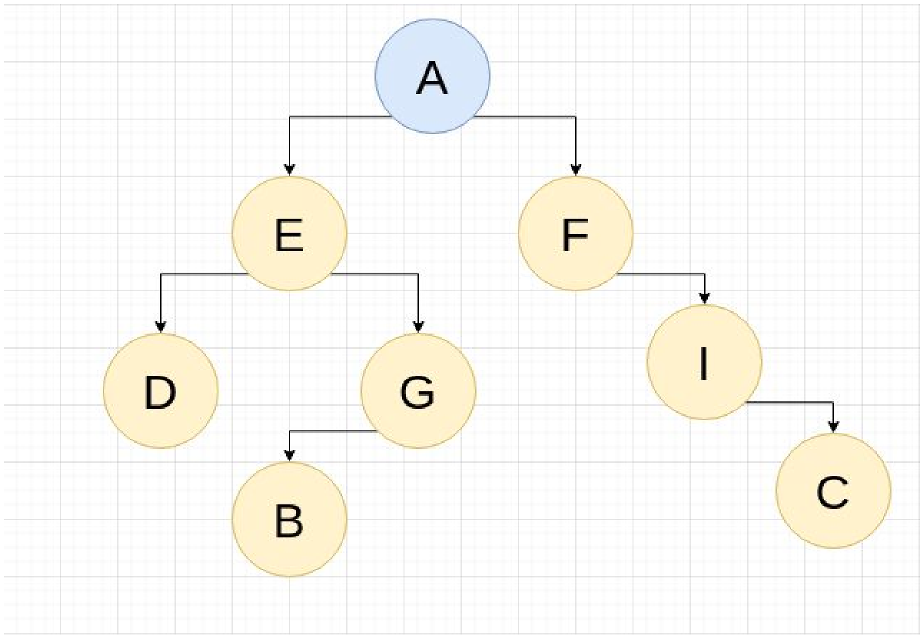
https://github.com/TorNATO-PRO/CptS-233-HW-2.git

Nathan Waltz
CPT_S 233

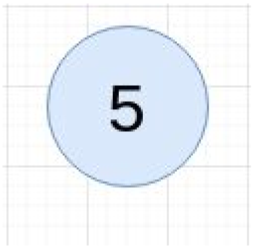
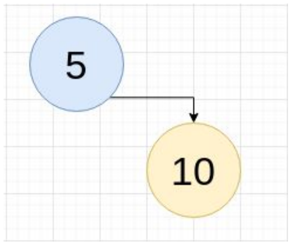
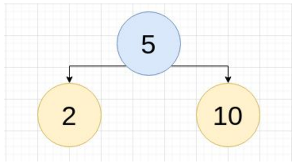
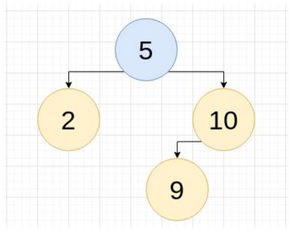
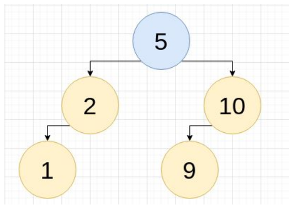
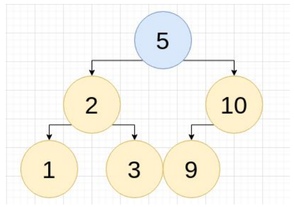
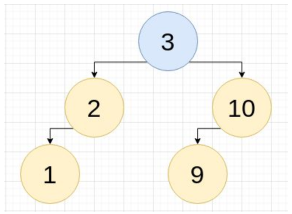
Homework #2

1. Given the following pre-order and in-order traversals, reconstruct the appropriate binary search tree. NOTE: You must draw a single tree that works for both traversals.

   Pre-order: A, E, D, G, B, F, I, C
   In-order: D, E, B, G, A, F, I, C

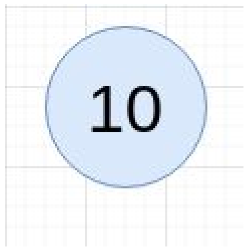
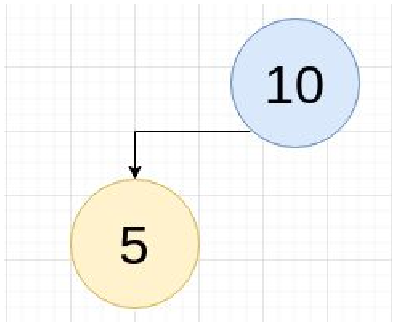
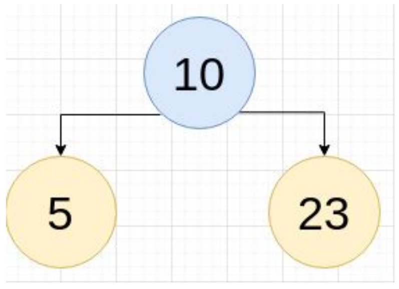
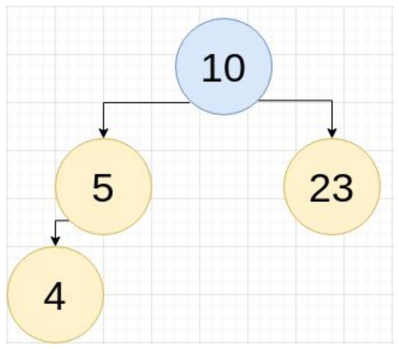
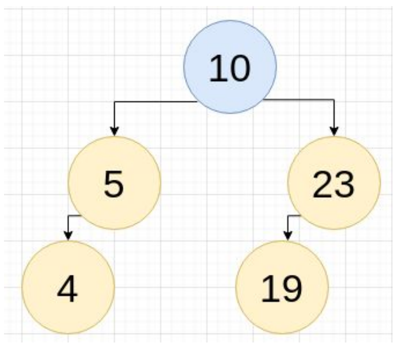
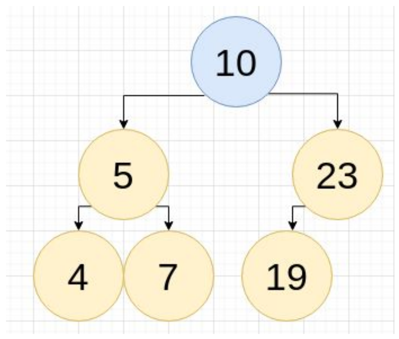


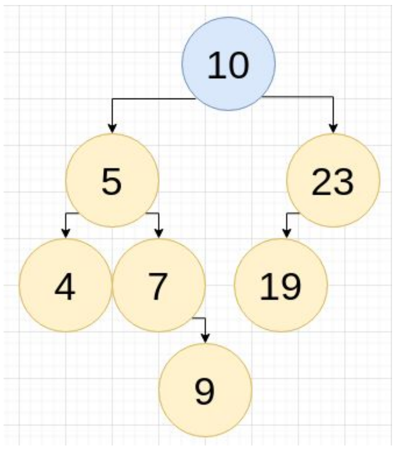
2. [3] Starting with an empty BST, draw each step in the following operation sequence. Assume that all removals come from the left subtree when the node to remove has two children.

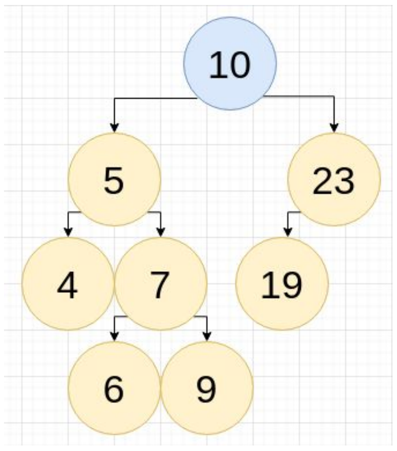
Insert(5) 	Insert(10) 	Insert(2) 	Insert(9) 
Insert(1) 	Insert(3) 	Remove(5) 	

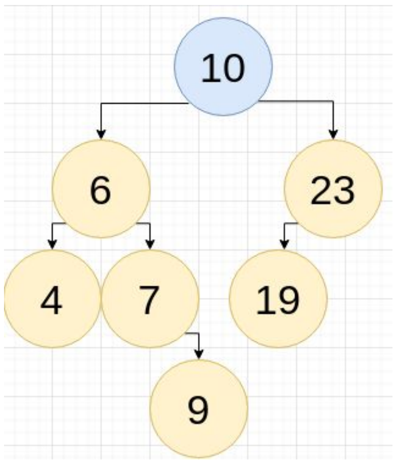
3. [3] Starting with an empty BST, draw each step in the following operation sequence. Assume that all removals come from the right subtree when the node to remove has two children.

Insert(10), Insert(5), Insert(23), Insert(4), Insert(19), Insert(7), Insert(9), Insert(6), Remove(5).

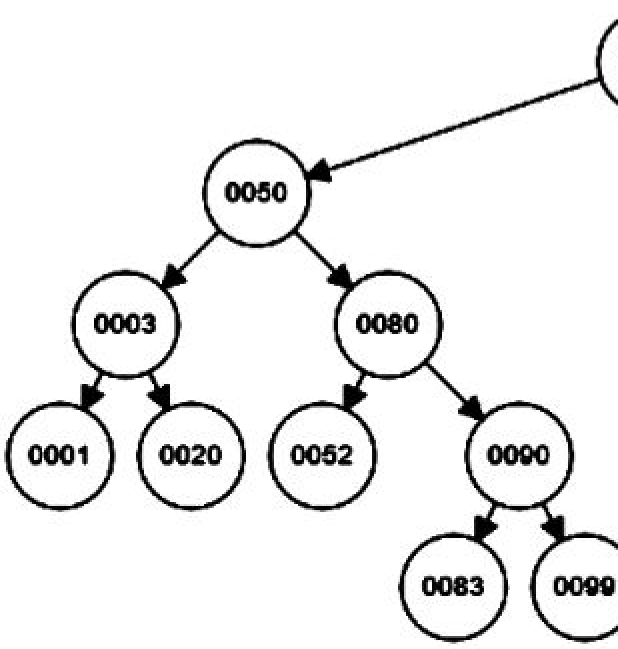
Insert(10) 	Insert(5) 	Insert(23) 
Insert(4) 	Insert(19) 	Insert(7) 
Insert(9)	Insert(6)	Remove(5)







4. Given the following binary tree (where nullptr height == -1):



a. [1] What is the height of the tree

Height would be 4 :)

b. [1] What is the depth of node 90?

The depth of node 90 would be 3

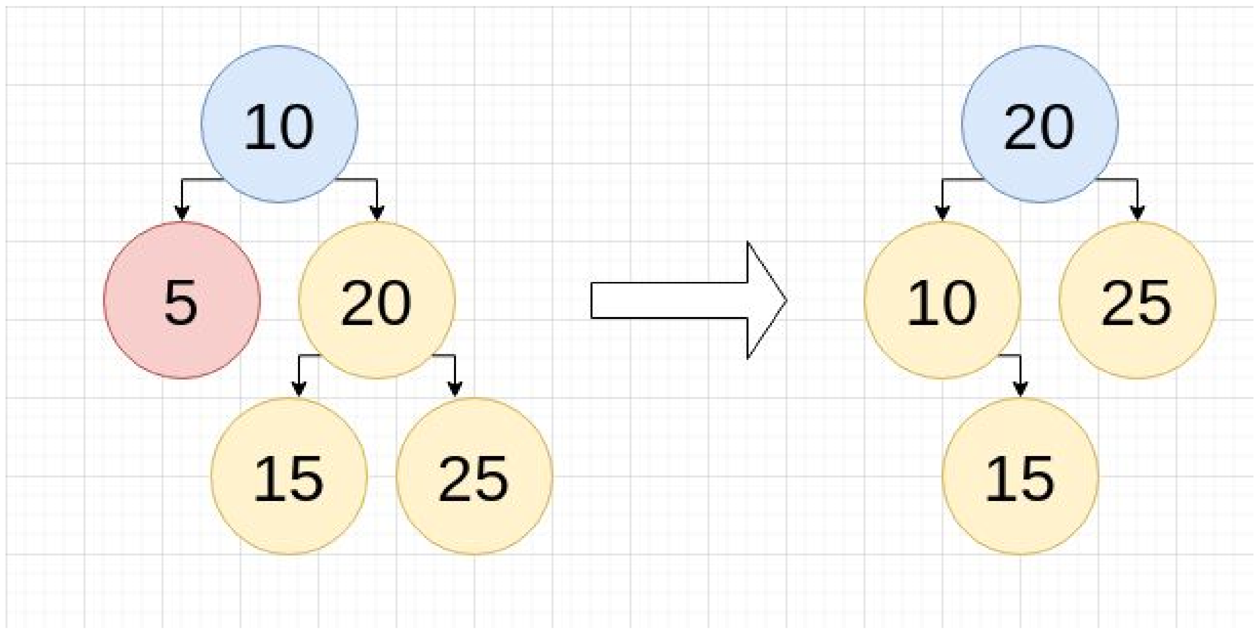
c. [1] What is the height of node 90?

The height of node 90 would be 1

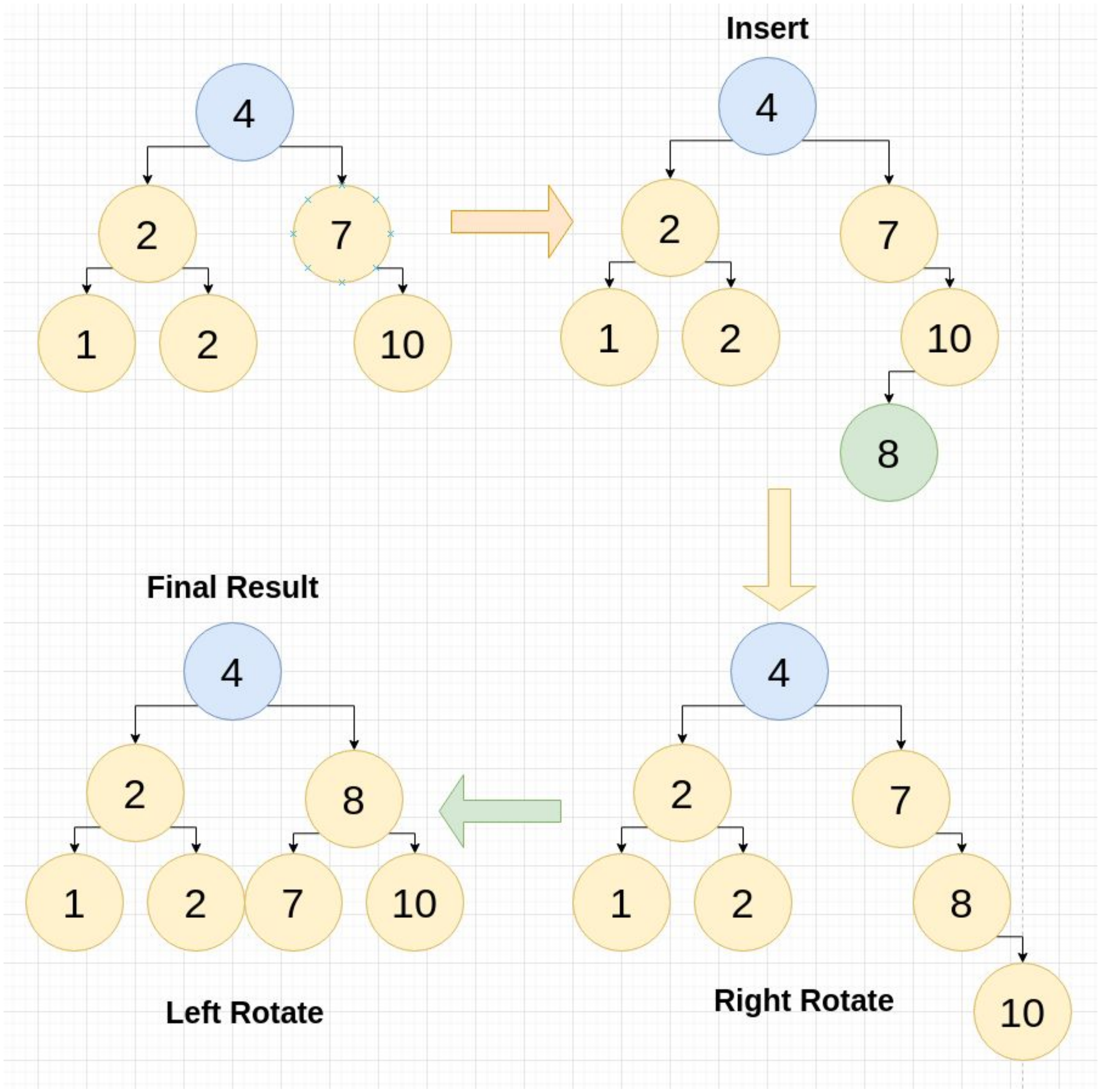
d. [3] Give the pre-order, in-order, and post-order traversal of this tree.

Pre-Order: 100, 50, 3, 1, 20, 80, 52, 90, 83, 99, 150, 125, 152
In-Order: 1, 3, 20, 50, 52, 80, 83, 90, 99, 100, 125, 150, 152
Post-Order: 1, 20, 3, 52, 83, 99, 90, 80, 50, 125, 152, 150,
100

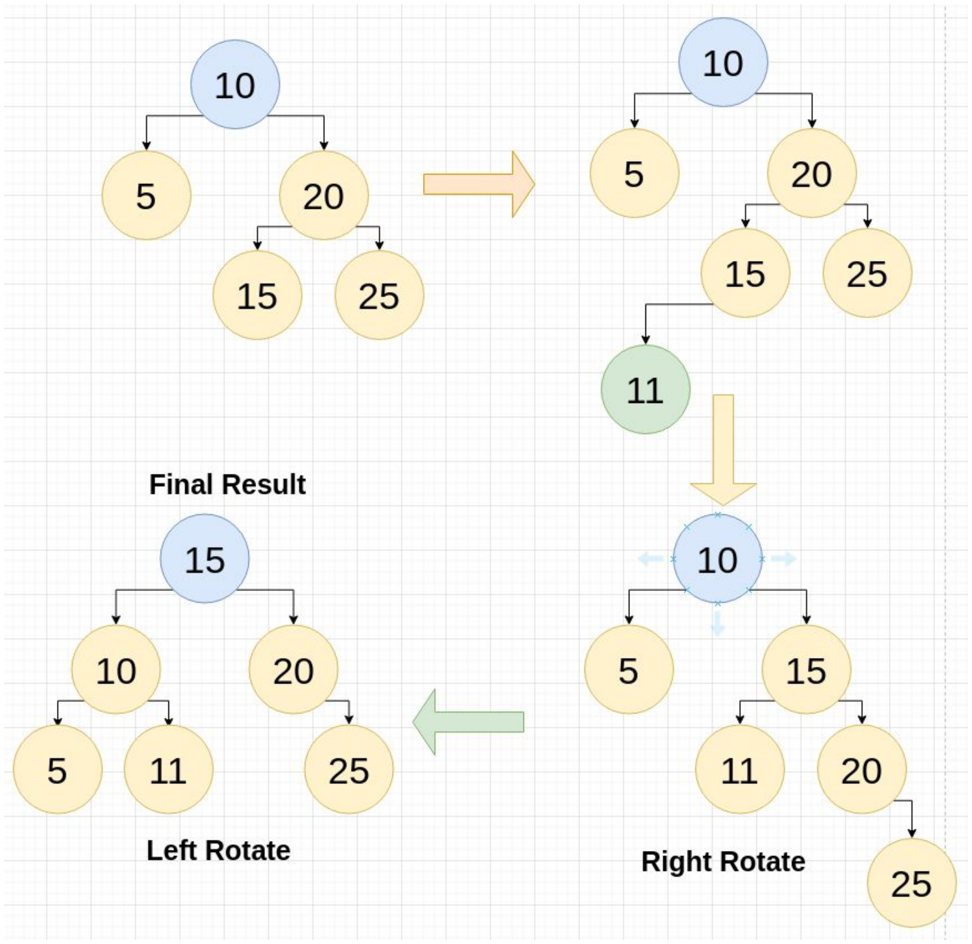
5. [3] Remove 5 from the following AVLtree; draw the results:



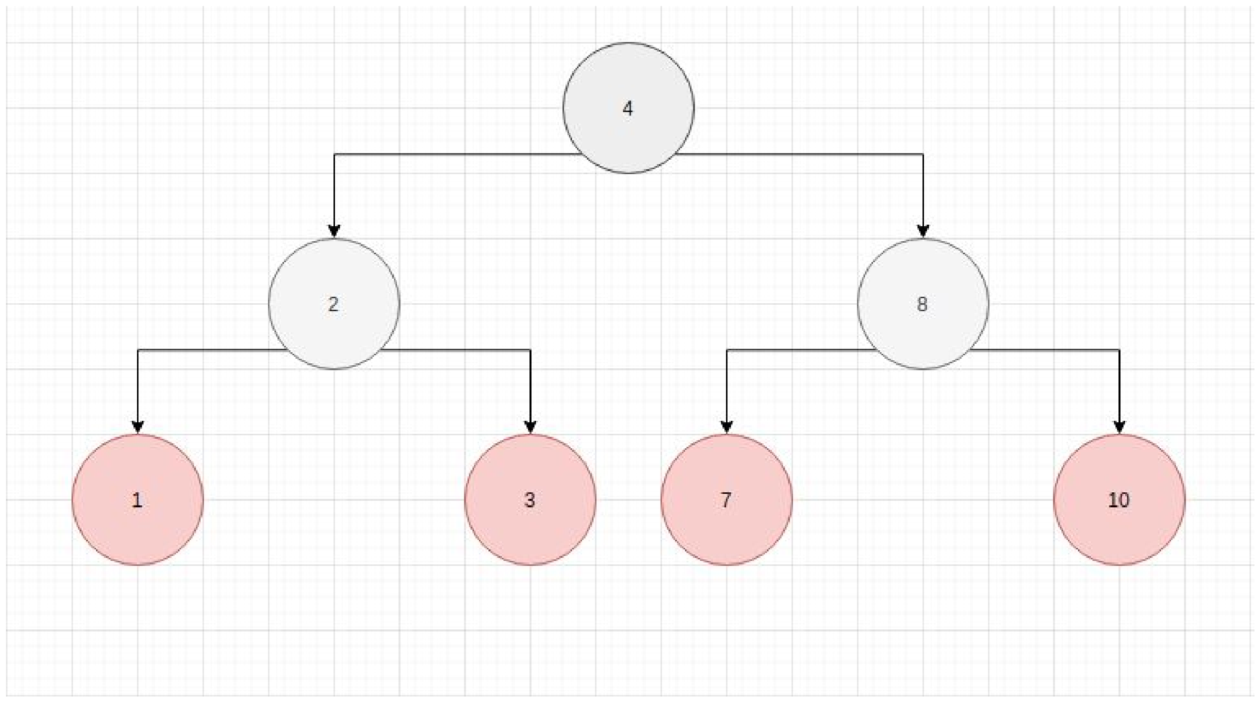
6. [3] Insert the value "8" into the following AVL tree; draw the result:



7. [3] Insert the value “11” into the following AVL tree; draw the result:

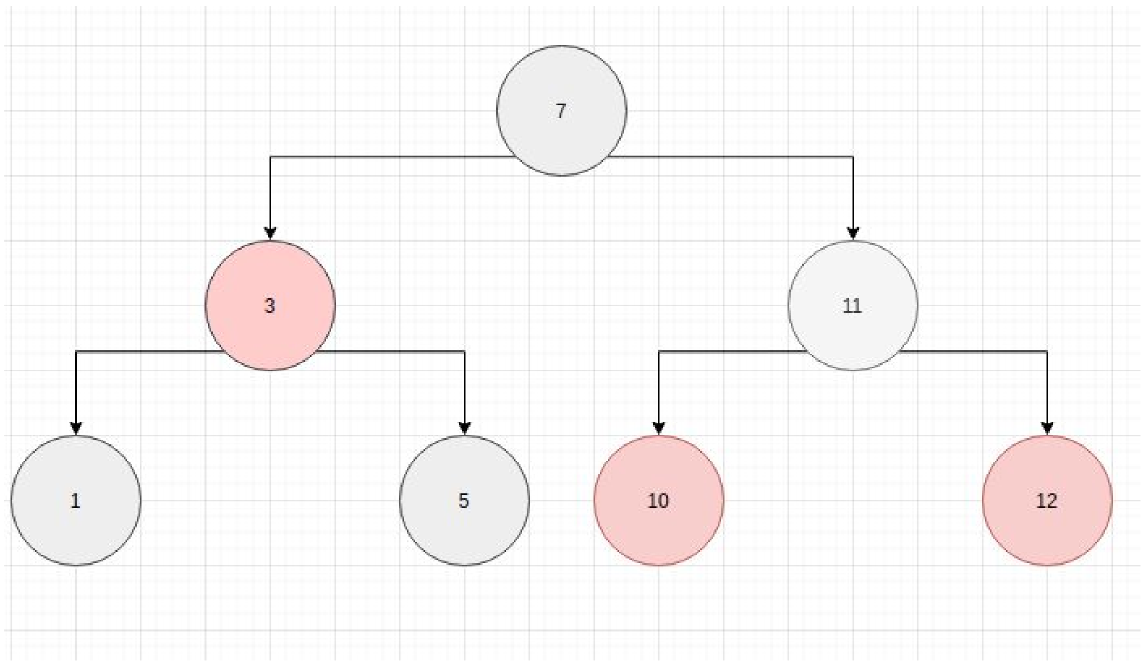


8. [3] Insert the value "8" into the following Red-Black tree; draw the result. Use Double-circle to denote red nodes and single circle to denote black nodes.



Grayish is black, red is red :)

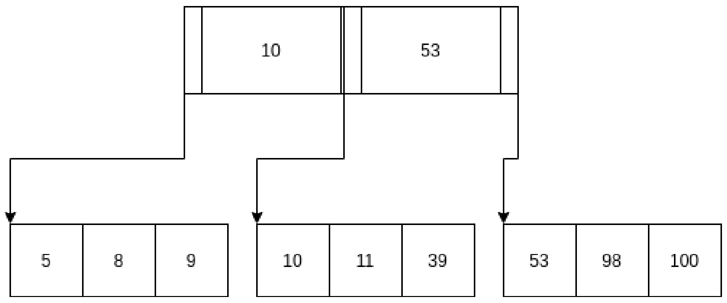
9. [3] Delete the value "2" from the following Red-Black tree; draw the result. Use Double-circle to denote red nodes and single circle to denote black nodes.



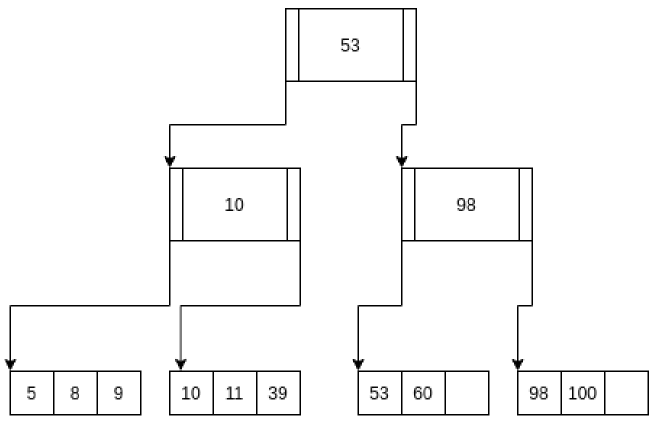
10. [4] Given the following B+ tree (M = 3, L = 3):

A) Insert 60 into the tree and draw the resulting B+ Tree:

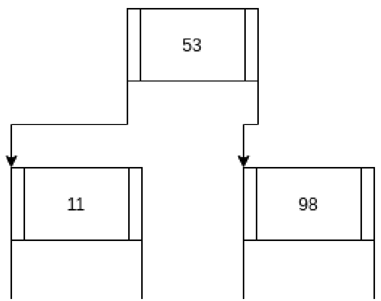
B) Based on the tree resulting from part (A), now remove 10 and draw the new tree:



Insert 60



Delete 10



11. [6] We are going to design our B+ Tree to be as optimal as possible for our old hard drives (since the management won’t buy new ones, those cheapskates!). We want to keep the tree as short as we can andpack each disk block in the filesystem as tightly as possible. We also want to access our data in sorted order for printing out reports, so each leaf node will have a pointer to the next one. See figure #1 on the next page for a visualization of our tree.

CPU architecture: Intel Xeon with 64 bit cores
Filesystem: Ext4 with 4KB (4096 byte) blocks
The customer records are keyed by a random UUID of 128 bits
Customer’s Data record definition from the java file:

```
#include <uuid>
struct CustomerData {
     UUID uuid;// Customer 128 bit key[1]
     char[32] name;// Customer name (char is 2 bytes each)[2]
     int ytd_sales;// Customer year to date sales};
}
```

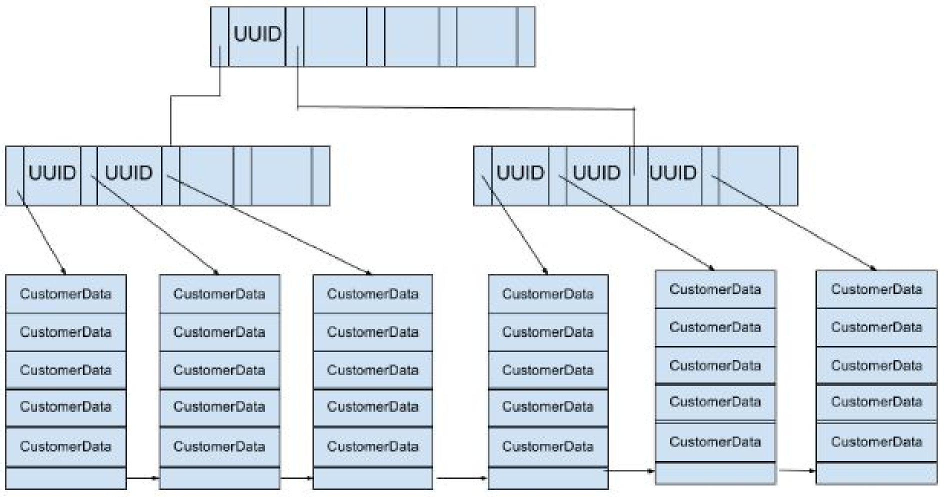


Figure #1: Visualization of our B+ Tree of height 2, customer data

a. Calculate the size of the internal nodes (M) for our B-tree:

Alrighty, so we know that this tree is m = 5, as there are 4 keys and 5 pointers. Given this information, we can calculate the sizes of the internal nodes!

$$numKeys \times sizeKeys \ + \ numPointers \times sizePointers \ = \ size\ of\ internal\ nodes\ (M)$$

$$4 \times 128\ bits + 5 \times 64\ bits \ = 832\ bits \ = 104\ bytes$$

b. Calculate the size of the B-tree leaf nodes (L) for this tree make sure to include the pointer (note CPU architecture!) to keep the list of leaf nodes:

For the last one, were we not supposed to include the size of the pointers? Because it looked like we were. If so, please disregard the size of the pointers that I included inside of my calculation.

We know that the leaf are of length 5, and that they have a single pointer that points to the next leaf node. From this information, a value can be deduced regarding the size of the B-Tree leaf nodes (L).

$$sizeOfCustomerData \ = \ 32 \times 2\ bytes + \ \frac{128\ bits}{8\ bits/byte} + 4\ bytes = 84\ bytes$$
$$sizeOfCustomerData \times 5\ (number\ of\ things) = 84 \times 5 \ = 420\ bytes$$
$$420\ bytes \ + 8\ bytes\ (size\ pointer) \ = \ 428\ bytes$$

c. How tall (on average) will our tree be (in terms of M) with N customer records?

So, with 5 customer records, the height will just be 0. For 25 customer records (remember 5 pointers), the height at minimum will be 1. For 125 customer records, the height at minimum will be 2. So,

$log_5(N) - 1 \leq Height_M$

d. If we insert 30,000 CustomerData records, how tall will the tree be?

$log_5(N) - 1 \leq Height_M$

$log_5(30,000)\ -\ 1 = 6\ \leq Height_M$

e. If we insert 2,500,000 customers how tall will the tree be?

$log_5(2,500,000) -\ 1 = 9 \leq Height_M$