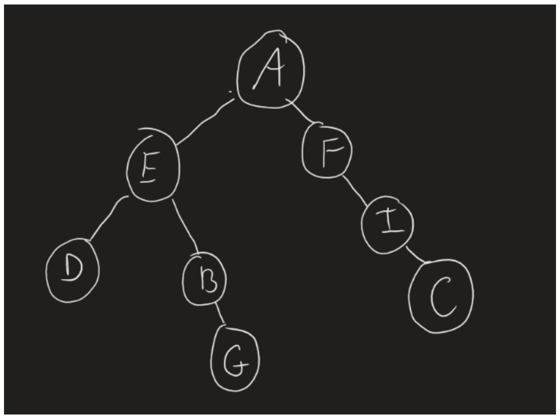
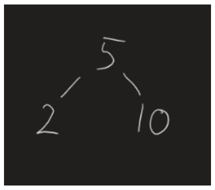
Cpts 223 – Nadra Guizani
Homework #2 – All About Those Trees
Tay Ho

1. [3] c **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

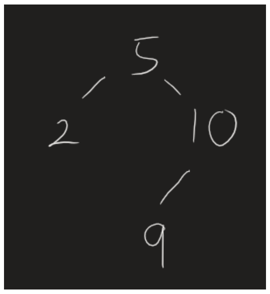


2. [3] Starting with an empty BST, draw each step in the following operation sequence. Assume that all removals come from the left subtree when the node to remove has two children.
Insert(5), Insert(10), Insert(2), Insert(9), Insert(1), Insert(3), Remove(5).

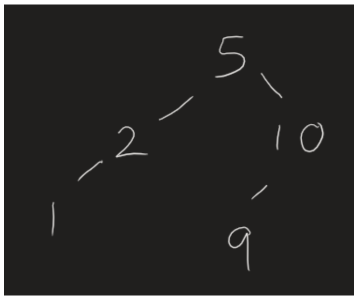
Insert 5 as root. Then, insert 10 to the right of 5 (10>5). Insert, 2 to the left of 5 (2<5).



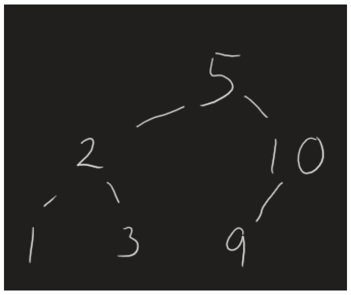
Insert (9) and because 9 > 5, and 9 < 10, 9 to the left of 10.



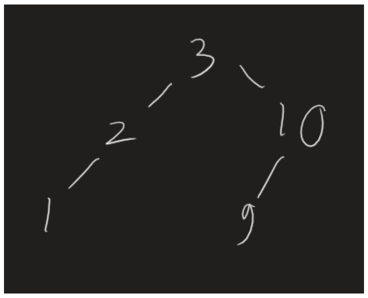
Insert 1 to the left of 2 (1 < 2).



Insert 3 to the right of 2 (2< 3)

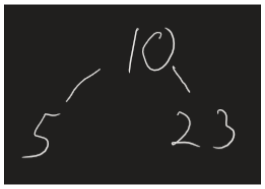


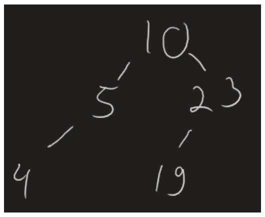
Remove (5). Since all removals come from the left subtree and 5 is the deleted root, 3 will be the next root (look for biggest value in left child and replace 5).

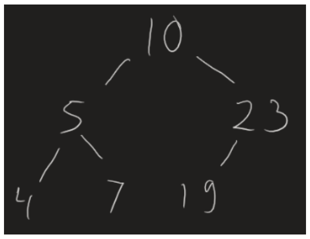


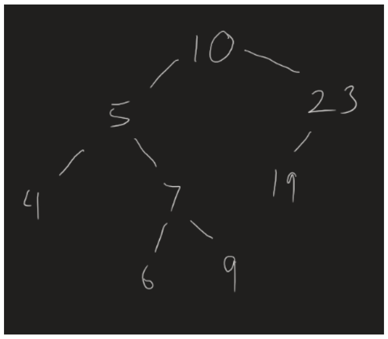
```
3. [3] Starting with an empty BST, draw each step in the following operation
sequence. Assume that all removals come from the right subtree when the node
to remove has two children.
```

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

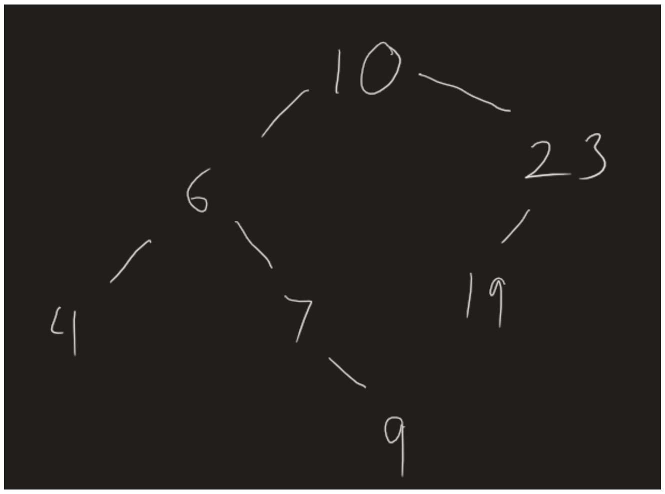




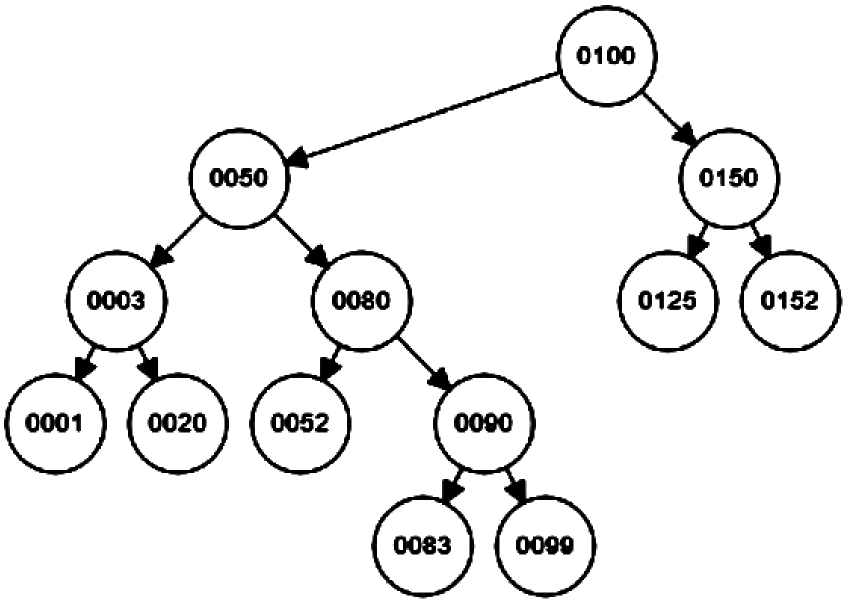




Remove 5. Look for the smallest value in right child to replace 5.



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



A. [1] What is the height of the tree?   Height is 4.

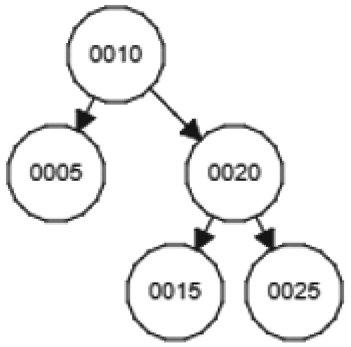
B. [1] What is the depth of node 90?   Depth of node 90 is 3.

C. [1] What is the height of node 90?   Height of node 90 is 2.

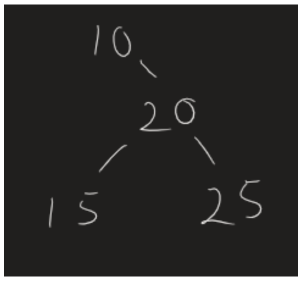
D. [3] Give the pre-order, in-order, and post-order traversal of this tree.

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

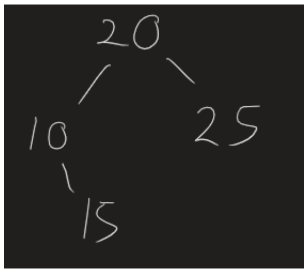
5. [3] Remove 5 from the following AVL tree; draw the results:



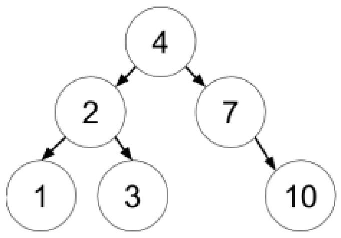
When remove 5 the tree is unbalance (balance factor = -2).

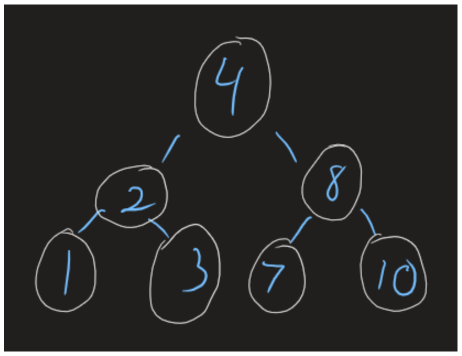


The algorithm than will do left rotation to bring the tree back to balance.

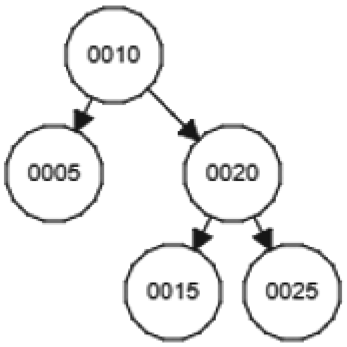


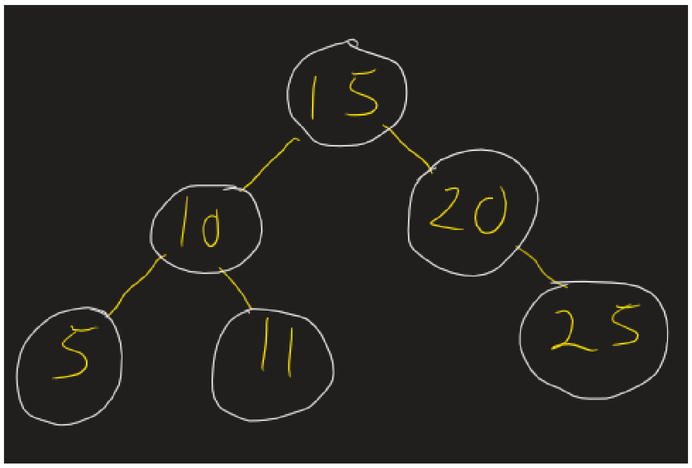
6. [3] Insert the value "8" into the following AVL tree; draw the result:



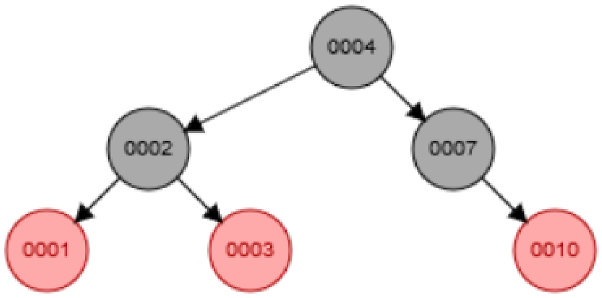


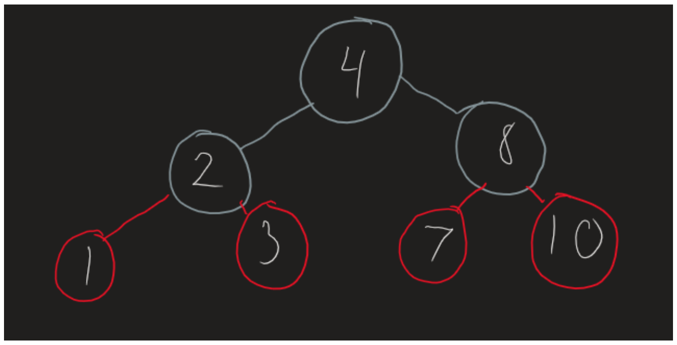
7. [3] Insert the value “11” into the following AVL tree; draw the result:



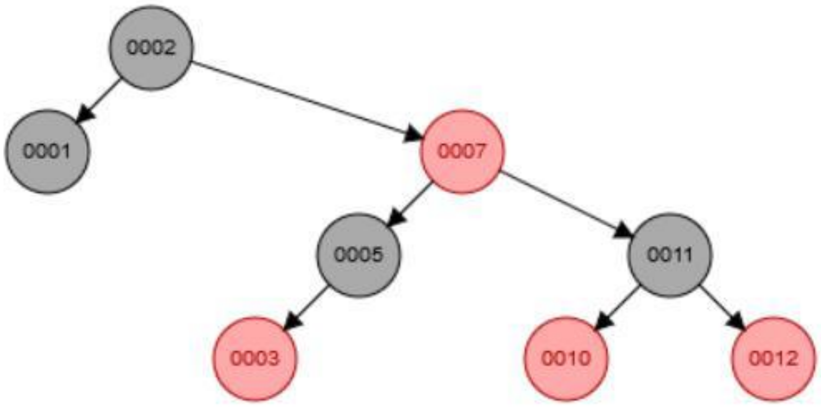


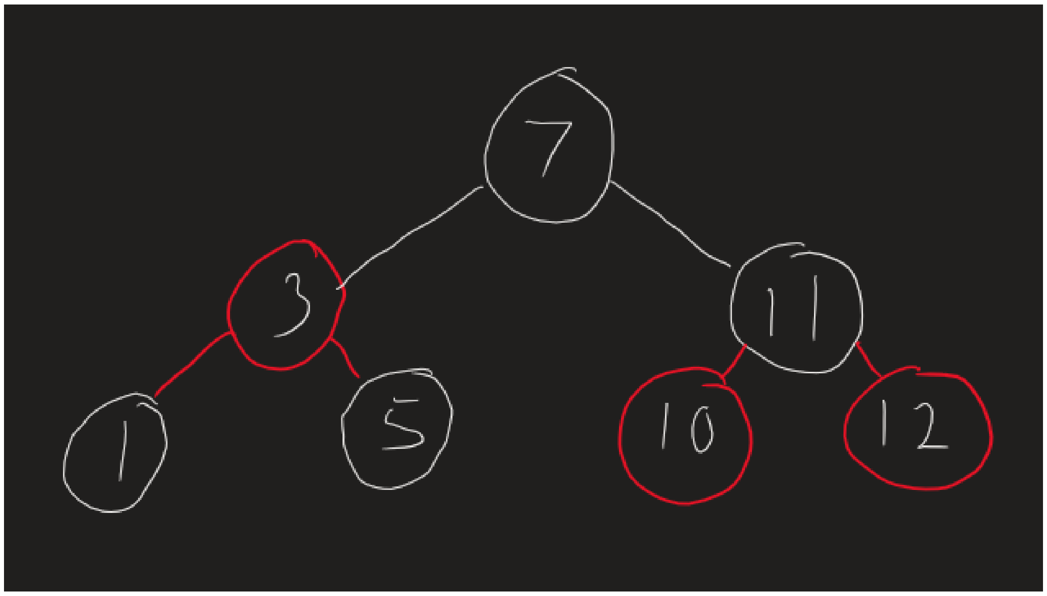
8. [3] Insert the value "8" into the following Red-Black tree; draw the result. Use Double-circle to denote red nodes and single circle to denote black nodes.



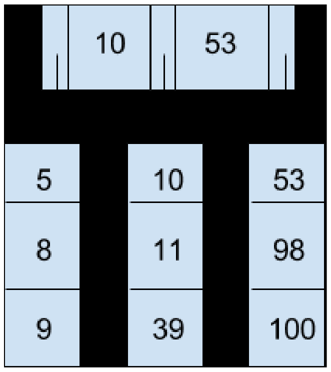


9. [3] Delete the value "2" from the following Red-Black tree; draw the result. Use Double-circle to denote red nodes and single circle to denote black nodes.

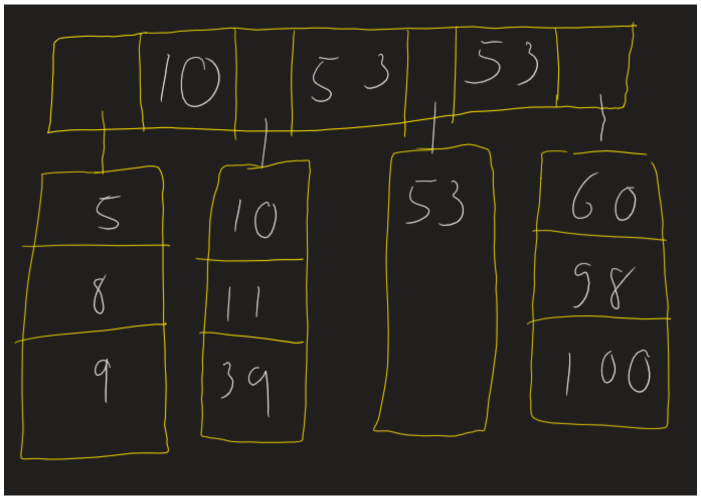




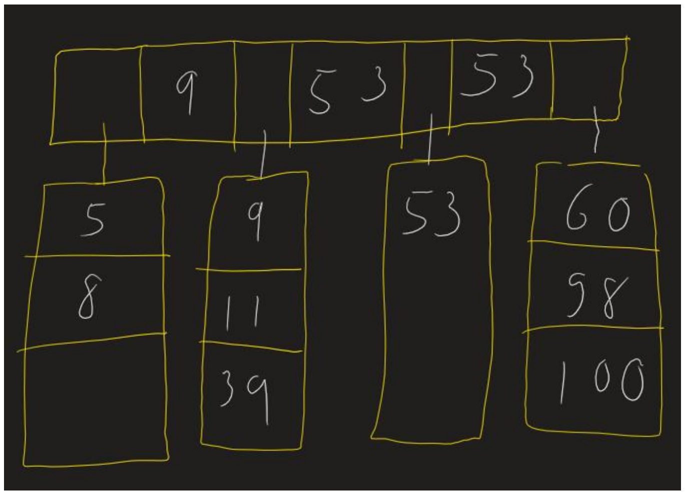
10. [4] Given the following B+ tree (M = 3, L = 3):



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:



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 and pack 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 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 key1
	char[32] name; // Customer name (char is 2 bytes each)2
	int ytd_sales; // Customer year to date sales
	};
```

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

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:

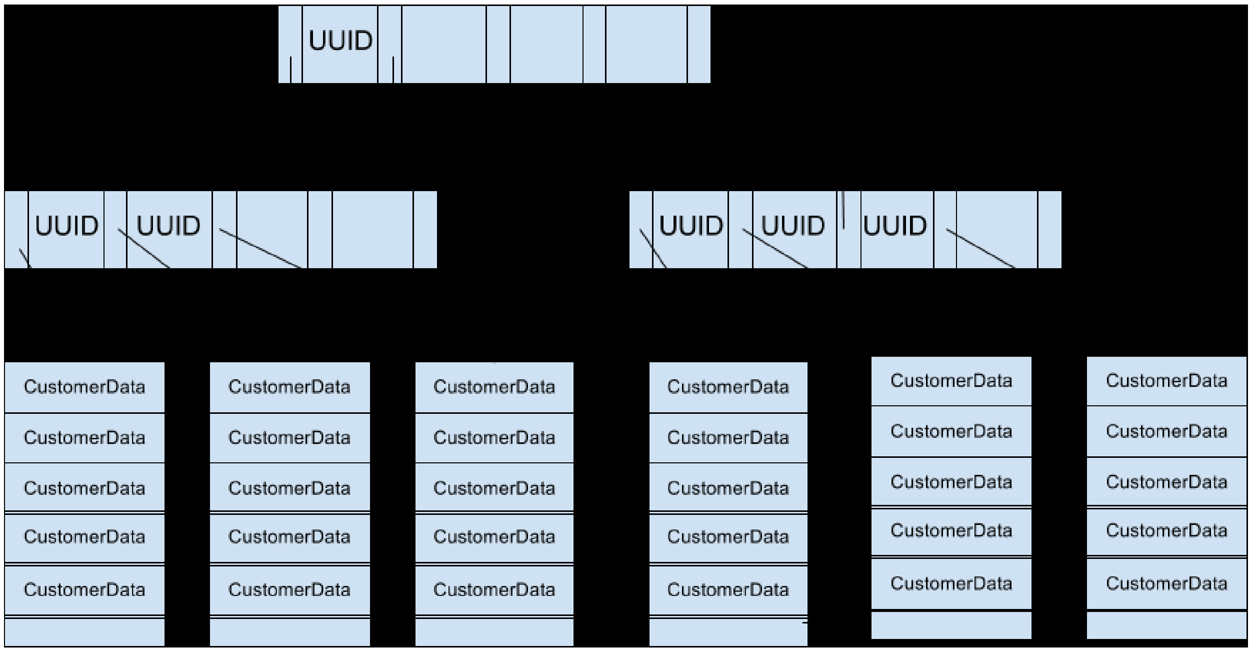


Figure #1: Visualization of our B+ Tree of height 2, customer data records, and pointers between the leaf nodes.

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

```
The B+ tree node can hold 5 points and up to 25 record.
5^m < n <= 5^(m+1)   in height (m)
```

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

```
5^6 < 30,000 <= 5^7
15,625 < 30,000 <= 78125
So the height of the tree will be 6.
```

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

5^9 < 2,500,00 < 5^10

1,953,125 < 2,500,00 < 9,765,625

So height will be 9.