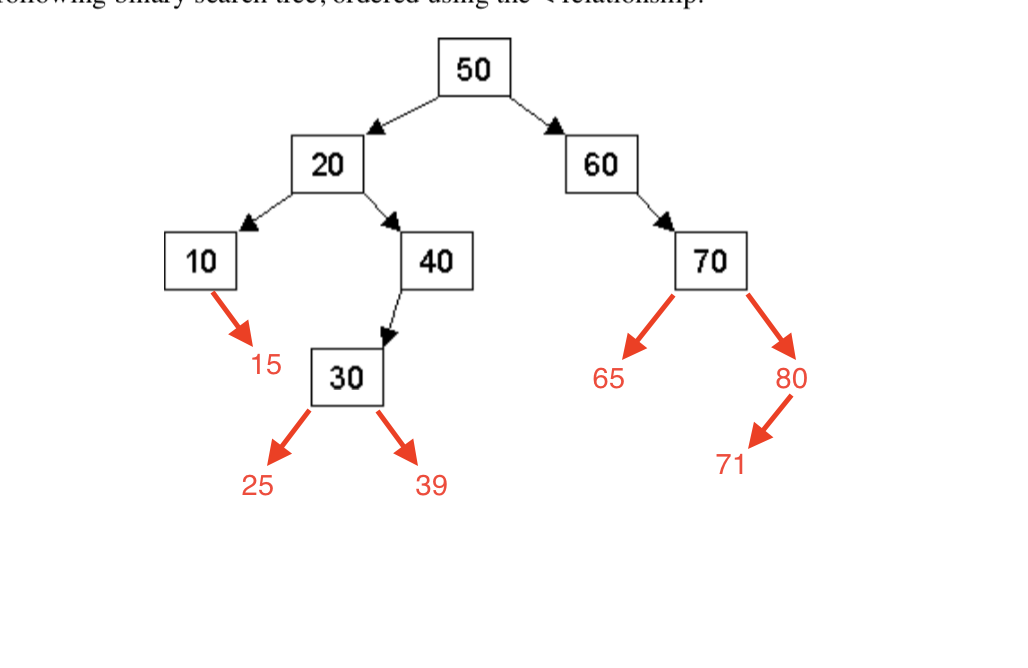
William Randall

UID: 805167986

1.

a.

  
B.

After inserting the nodes mentioned in part a, what would be printed out by in-order, pre-order, and post-order traversals of the tree (assume your traversal function prints out the number at each node as it is visited)?

In-order:

10,15,20,25,30,39,40,50,60,65,70,71,80

Pre-order:

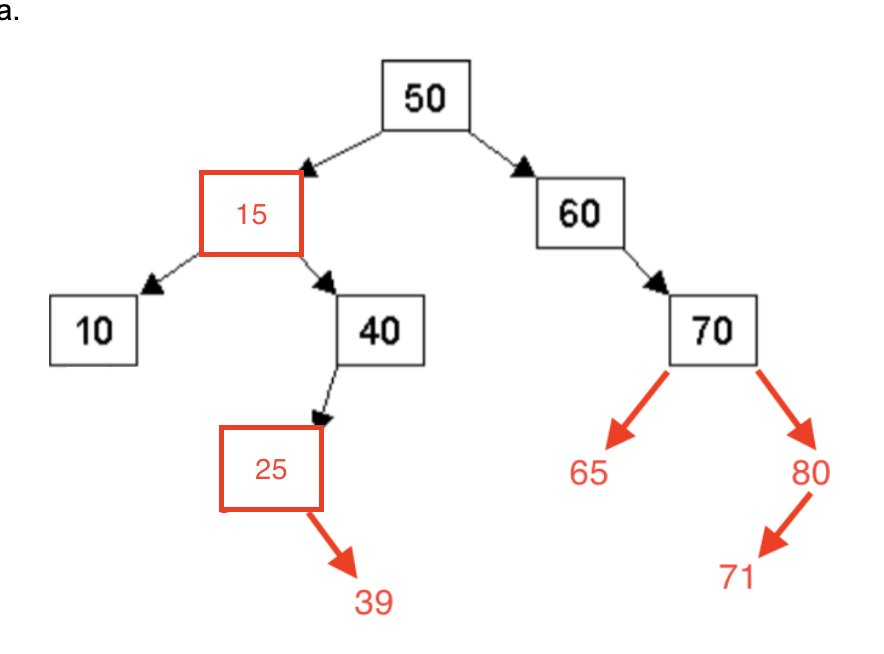
50,20,10,15,40,30,25,39,60,70,65,80,71

Post-order:

15,10,25,39,30,40,20,65,71,80,70,60,50

C.

After inserting the nodes mentioned in part a, what is the resulting BST after you delete the node 30, then the node 20? (Again, just use a simple deletion algorithm with no balancing. If you have an option of making a choice, any correct choice is acceptable.)



2.

A.

Show a C++ structure/class definition for a binary tree node that has both child node pointers and a parent node pointer. Assume the data stored in each node is an int.

Struct Node

{

Node(int val, Node\* parent)

{

m\_left = nullptr;

m\_right = nullptr;

m\_parent = parent;

m\_val = val;

}

Node\* m\_left;

Node\* m\_right;

Node\* m\_parent;

Int m\_val;

};

B.

Write pseudocode to insert a new node into a binary search tree with parent pointers. (Hint: You can find binary search tree insertion code on pp. 471-473).

void help(Node \* current, int val,Node \* parent)

{

if your current points to nullptr

{

make a new node

set its value

set left to nullptr

set right to nullptr

set parent to parent node

return

}

else if data given is less than data at current node

{

insert(current->left, val, cur);

}

else

{

insert(current->right, val, cur);

}

}

void insert(int data, Node \* begin)

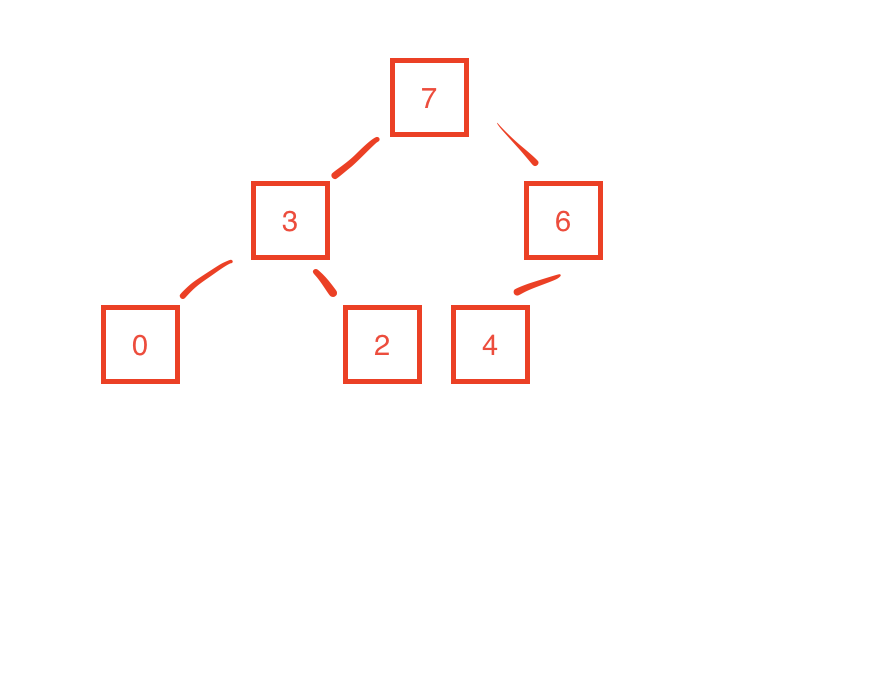
{

help(begin, val, nullptr);

}

3.

A.



B.

(7, 3, 6, 0, 2, 4)

C.

(6, 3, 4, 0, 2)

4.

1 . O( C + S )

2 . O( logC + S)

3 . O( log( C S ) )

4 . O( logS )

5 . O( 1 )

6 . O( logC + S )

7 . O( S )

8 . O( C log S )