

BINARY SEARCH TREE

Problem Solving with Computers-II

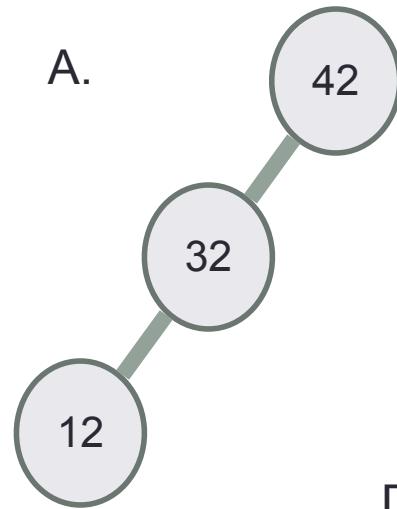
C++

```
#include <iostream>
using namespace std;

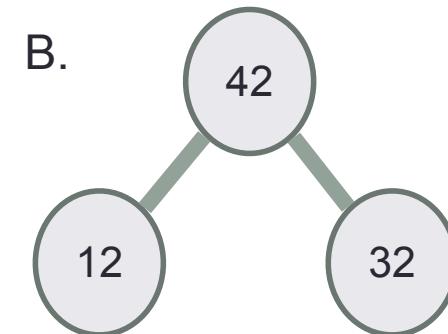
int main(){
    cout<<"Hola Facebook\n";
    return 0;
}
```

Which of the following is/are a binary search tree?

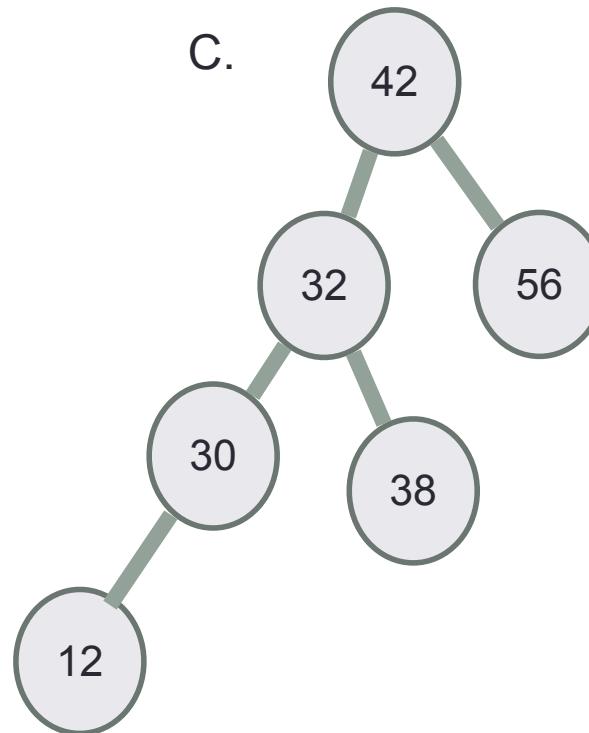
A.



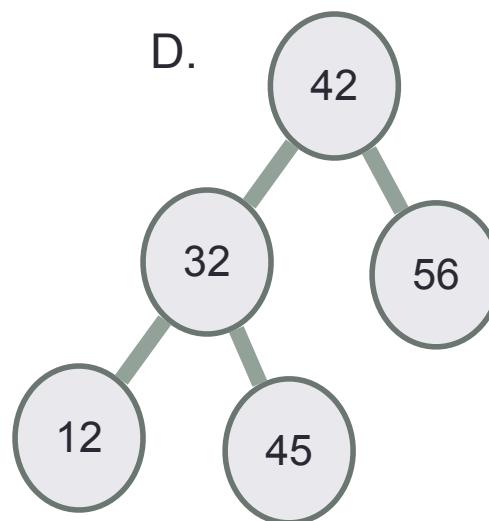
B.



C.



D.



E. More than one of these



Goal: To articulate the algorithm for inserting a key into a BST

Insert keys: 41, 45, 32, 42, 12

Traversal algo	BST	Output
<pre data-bbox="0 195 788 606">inorder(r: pointer to current node): if r is null: return inorder(r->left) process r (e.g., print r->val) inorder(r->right)</pre>		
<pre data-bbox="0 606 788 1031">postorder(r: pointer to current node): if r is null: return postorder(r->left) postorder(r->right) process r (e.g., print r->val)</pre>		
<pre data-bbox="0 1031 788 1440">preorder(r: pointer to current node): if r is null: return process r (e.g., print r->val) preorder(r->left) preorder(r->right)</pre>		

Post-order traversal: use to recursively clear the tree!

postorder(r : pointer to current node):

```
if r is null, return  
postorder(r->left)  
postorder(r->right)  
process r (e.g., print r->val)
```

```
int bst::getHeight(Node *r) const{  
    if (!r)  
        return -1;  
    int hleft = getHeight(r->left);  
    int hright = getHeight(r->right);  
    return max(hleft, hright) + 1;  
}
```

Why would preorder not work for clear?

```
void bst::clear(Node *r){  
    if (!r)  
        return;  
    clear(r->left);  
    clear(r->right);  
    delete r;  
}
```

When would you use each traversal and why?

Inorder: useful when. . .

Postorder: useful when. . .

Preorder: useful when. . .

Pre-order traversal Game!

1. **Draw a BST:** Individually, draw a BST with 6 distinct keys (e.g., integers 1–10). Your
Your BST (draw secretly):

2. **Trace Preorder Sequence:** Trace the preorder traversal (root, left, right) of your BST
and write the sequence of node values.

Your Preorder sequence:

(read the sequence to your partner, don't show them your BST diagram!)

Pre-order traversal Game!

3. Write the PreOrder sequence you received:

4. **Reconstruct the BST:** Using your partner's sequence, rebuild their BST by inserting nodes in the given order, respecting BST properties.

5. Compare trees

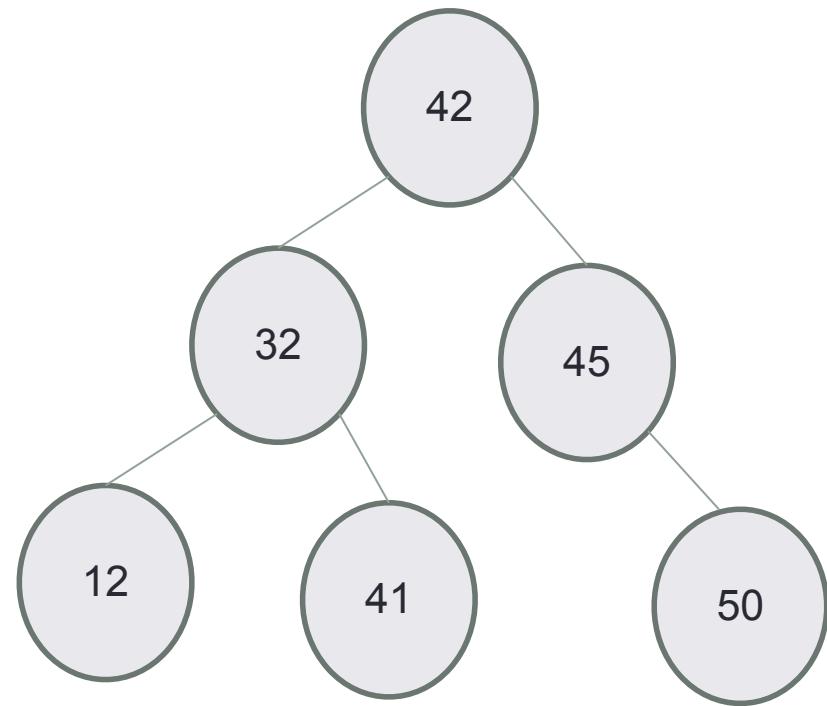
Leetcode problem

Given the root of a binary tree, return the preorder traversal of its nodes' values in a vector.

```
class Solution {  
public:  
    vector<int> preorderTraversal(TreeNode* root) {}  
};
```

<https://leetcode.com/problems/binary-tree-preorder-traversal/description/?envType=problem-list-v2&envId=depth-first-search>

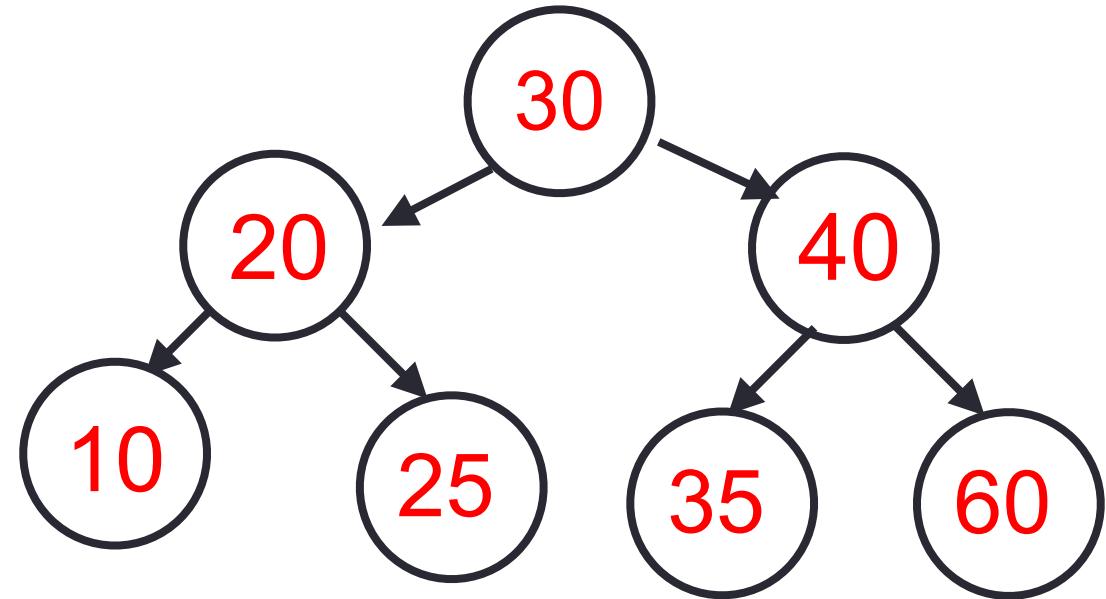
Interview question: Write a function to extract BST keys into a vector. The extraction order must allow reconstructing the exact same tree structure by sequential insertion.



<https://leetcode.com/problems/binary-tree-preorder-traversal/description/?envType=problem-list-v2&envId=depth-first-search>

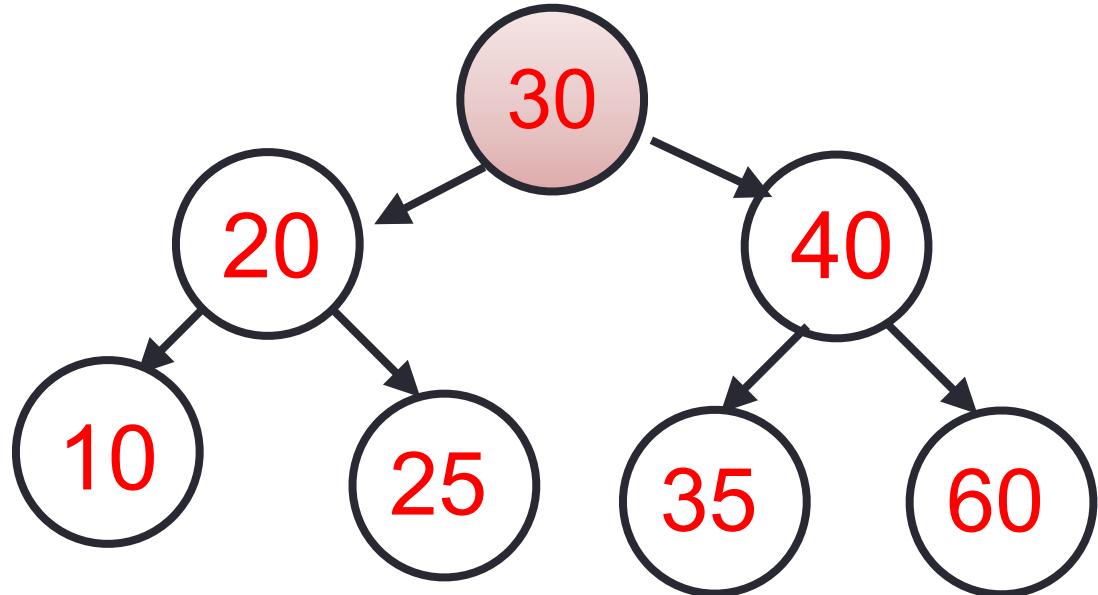
What does this code do?

```
Node* r = b.min(root);  
while(r){  
    cout << r->data << " ";  
    r = b.successor(r);  
}
```

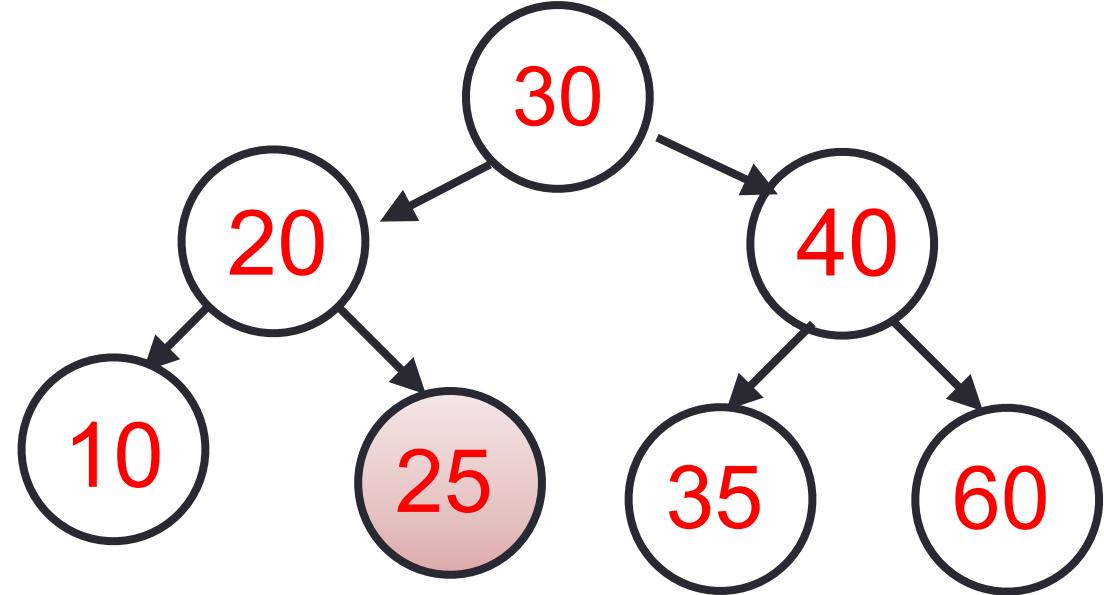


- What is the successor of 30?
- What is the successor of 25?

Successor: Next largest element

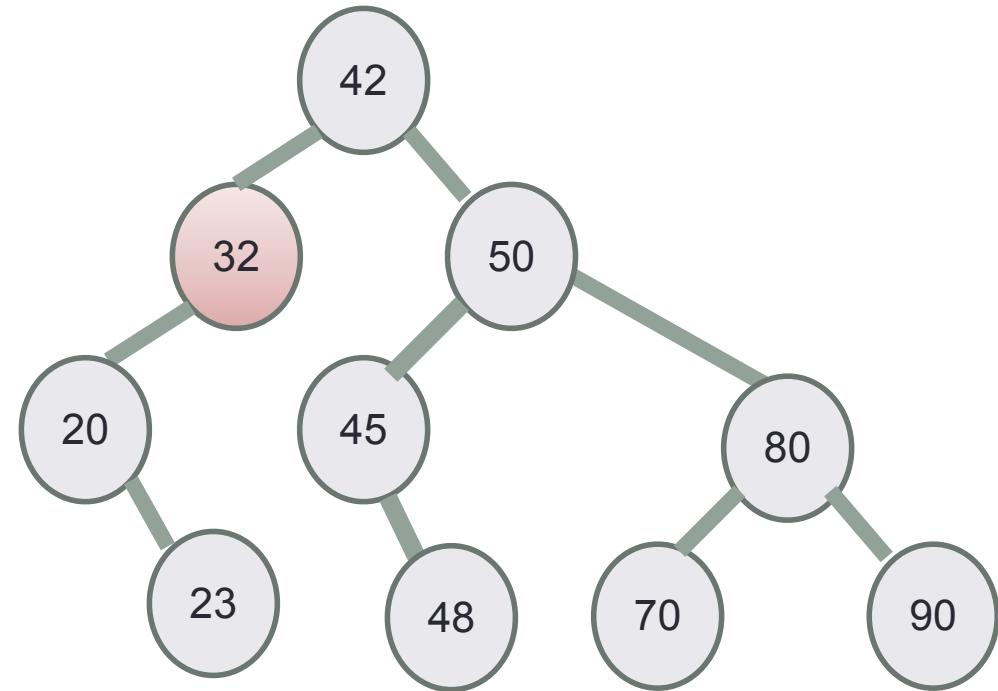
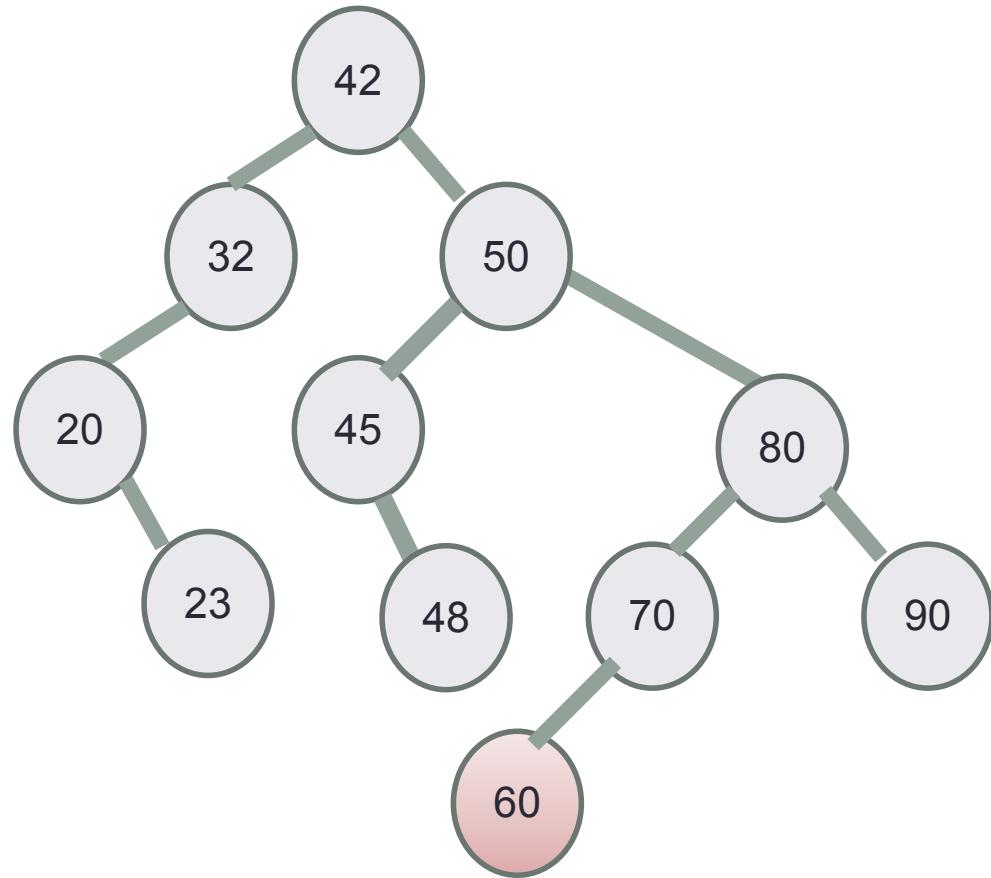


- Case 1: Node has a right child



- Case 2: Node has no right child

Delete a specific key value

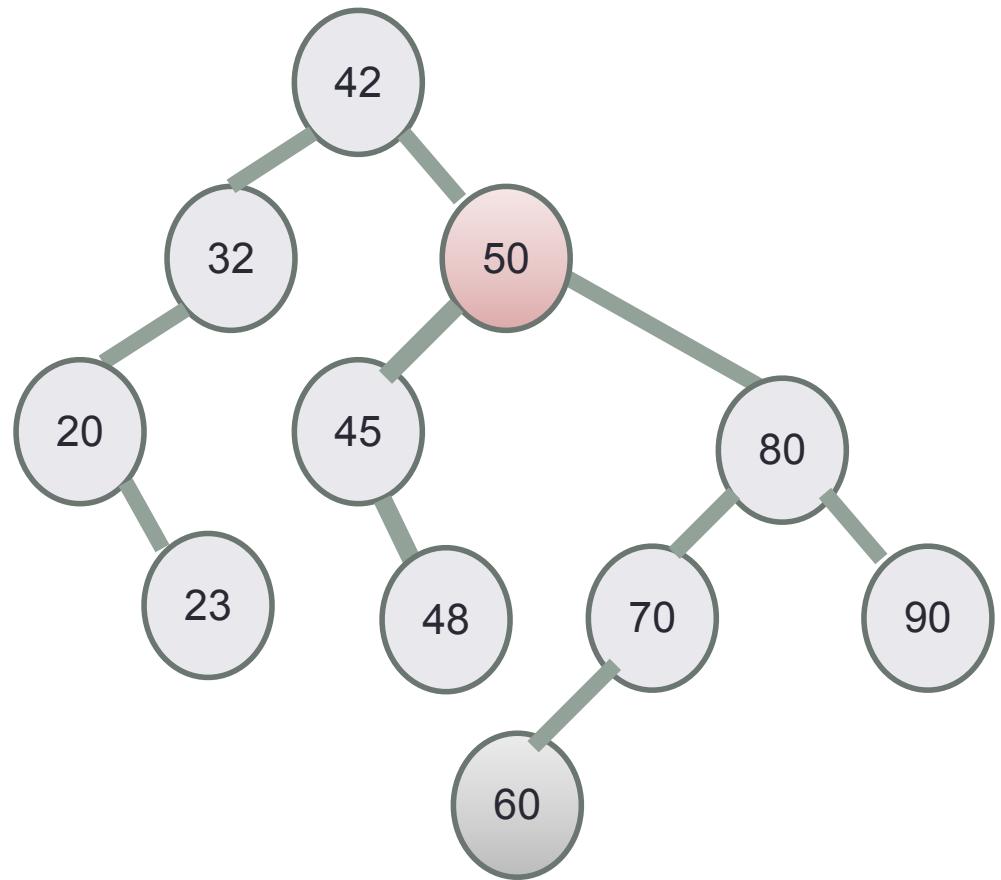


- Case 1: Node to delete is a leaf node
 - Set parent's (left/right) child pointer to null
 - Delete the node

- Case 2: Node has one child (left/right)
 - Replace the node by its only child

Delete a specific key value

- Case 3: Node has two children



BST ADT

Operations

Search

Insert

Min

Max

Successor

Predecessor

Delete

Print elements In order

Preorder,

Post order

