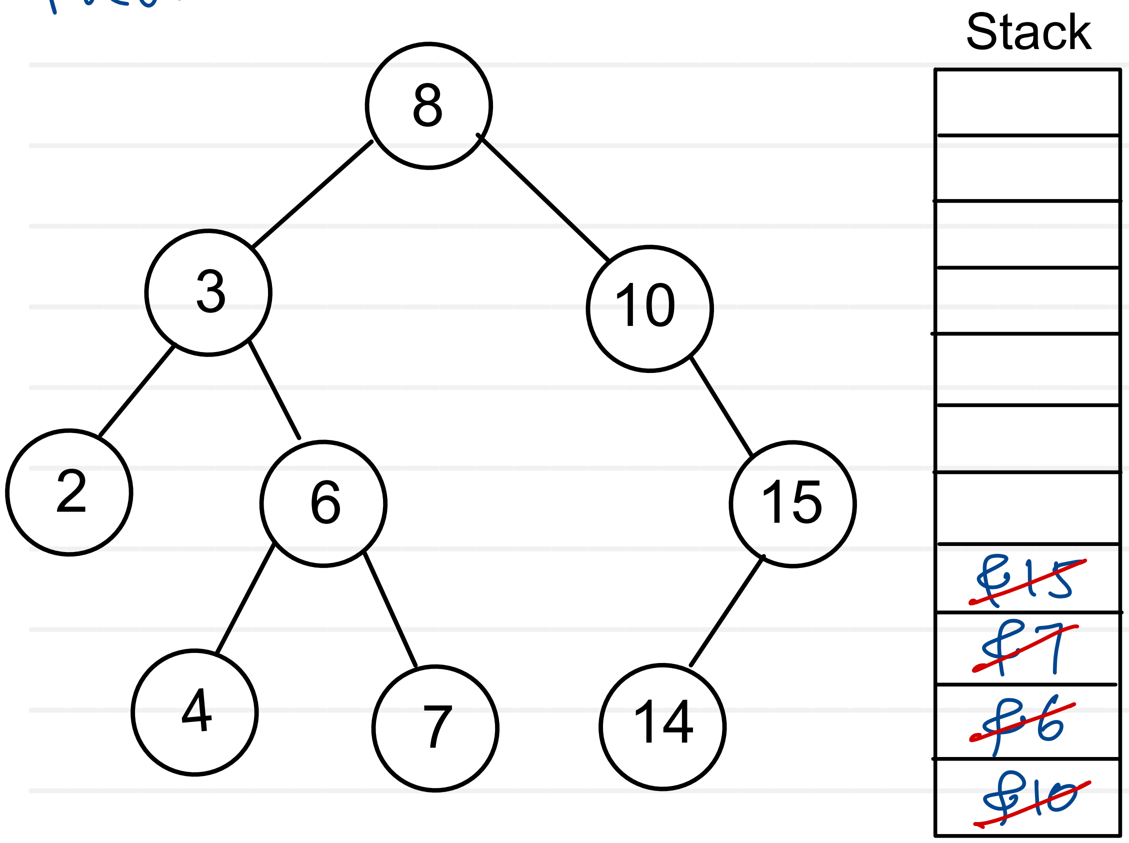
BST - Preorder (Non recursive)

Preorder = VLR

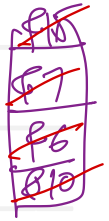


Preorder :

8 , 3 , 2 , 6 , 4 , 7 , 10 , 15 , 14

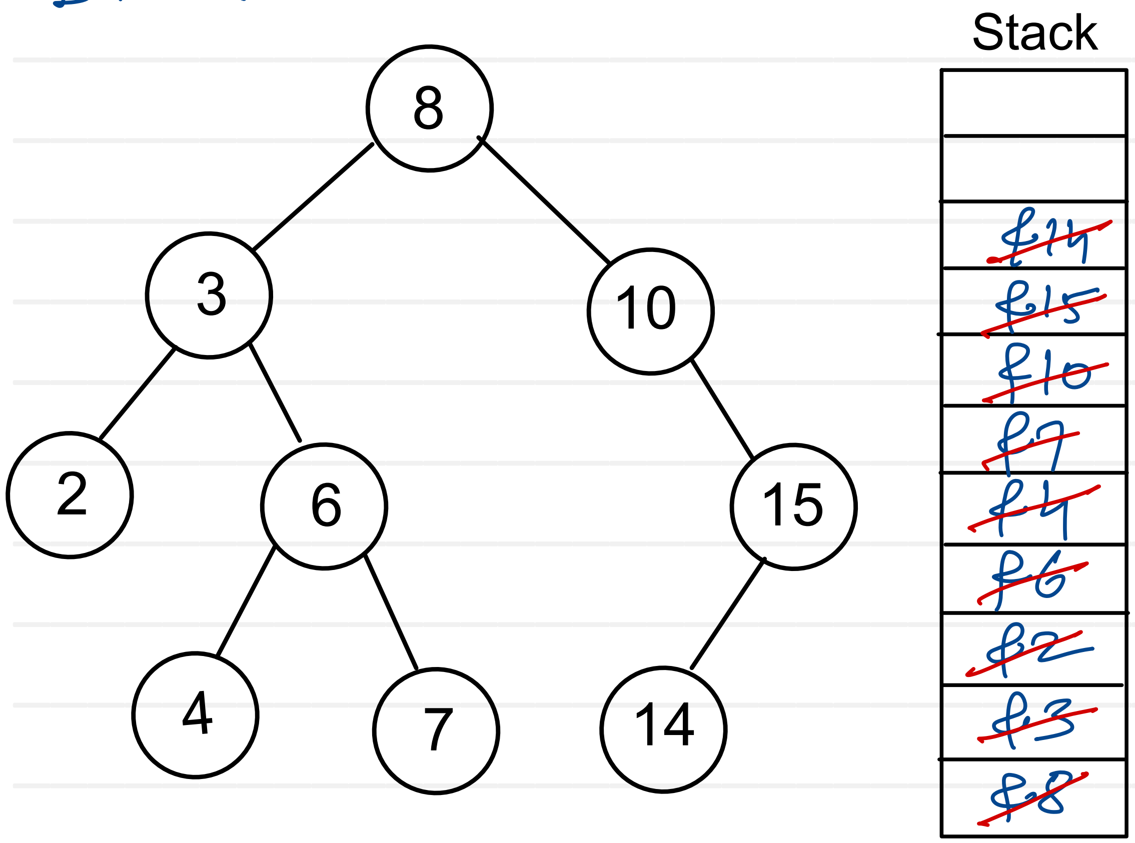
```
void preorder( ) {
    Stack<Node> st = new stack<>();
    //1. start traversing from root
    Node trav = root;
    while(trav != null || !st.empty()) {
        while(trav != null) {
            //visit current node,
            sysout(trav.data);
            // push right if exists on stack
            if(trav.right != null)
                st.push(trav.right);
            // go on left
            trav = trav.left;
        }// repeat till extreme left
        //2. pop node from stack.
        if( !st.empty())
            trav= st.pop();
    }
}
```

trav
&8
&6
&7
&10
&15



BST - Inorder (Non recursive)

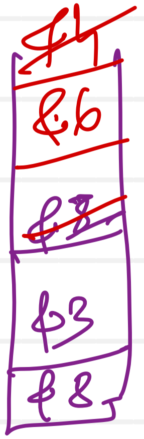
Inorder : LVR



Inorder:

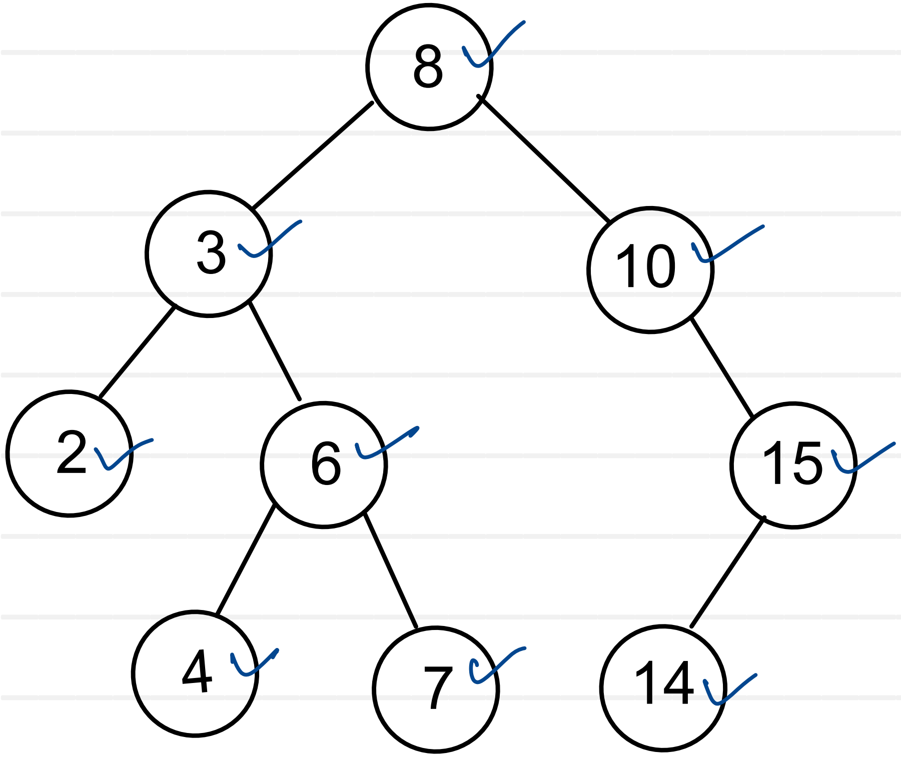
2, 3, 4, 6, 7, 8, 10, 14, 15

trav
&8
&3
&2
null



```
void inorder ( ) {
    Stack<Node> st = new Stack<>();
    //1. start from root
    Node trav = root;
    while(trav != null || !st.isEmpty()){
        // push node on stack & go into
           left, repeat till extreme left
        while(trav != null) {
            st.push(trav);
            trav = trav.left;
        }
        // pop element from stack, visit it
           & go into right.
        if(!st.isEmpty()) {
            trav = st.pop();
            sysout(trav.data);
            trav = trav.right;
        }
    }
}
```

BST - Postorder (Non recursive))



Stack

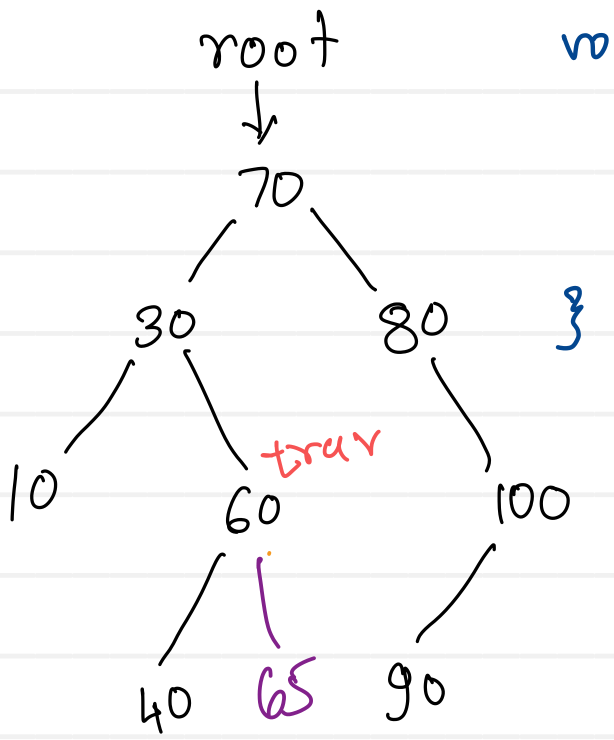
- ~~f14~~
- ~~f15~~
- ~~f10~~
- ~~f10~~
- ~~f8~~
- ~~f7~~
- ~~f6~~
- ~~f4~~
- ~~f6~~
- ~~f3~~
- ~~f2~~
- ~~f3~~
- ~~f8~~

```
void postorder( ) {
    stack<Node> st = new stack<>();
    // start from root
    Node trav = root;
    while(trav != null || !st.isEmpty()) {
        while( trav != null)
            st.push(trav);
            trav = trav.left;
        }
        if( !st.isEmpty()) {
            trav = st.pop();
            if(trav.right == null || trav.right.visited == true){
                sysout(trav.data);
                trav.visited = true;
                trav = null;
            } else {
                s.push(trav);
                trav = trav.right;
            }
        }
    }
}
```

Postorder:

2, 4, 7, 6, 3, 14, 15, 10, 8

Add Node (Recursion)



```
void add(int value) {
    if(root == null)
        root = new Node(value);
    else
        add(root, value);
}
```

```
add(65)
add(&70, 65)
add(&30, 65)
add(&60, 65)
```

```
void add(Node trav, int value) {
    if(value < trav.data) {
        if(trav.left == null)
            trav.left = new Node(value);
        else
            add(trav.left, value);
    }
    else {
        if(trav.right == null)
            trav.right = new Node(value);
        else
            add(trav.right, value);
    }
}
```