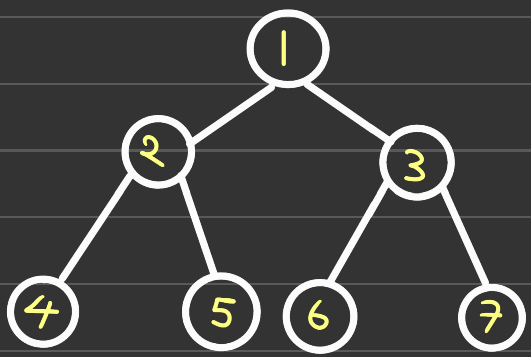
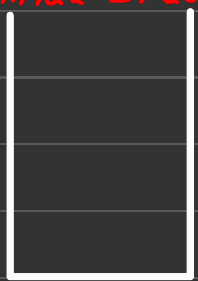
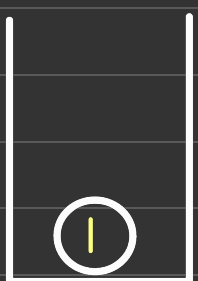
Pre ORDER (Iterative)



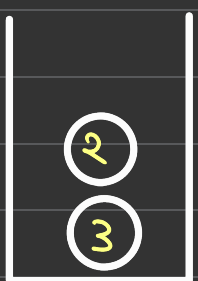
Initial Stack status!



1)



2)



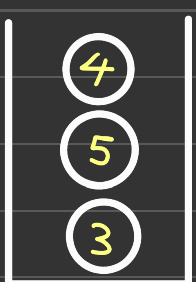
1 2 4 5 3 6 7

cout the node at the top of Stack And PoP it from Stack...

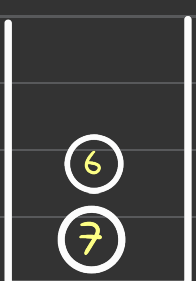
Also push its right node followed by left in the Stack...

Repeat while (stack.size()>0);

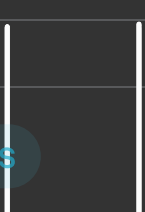
3)



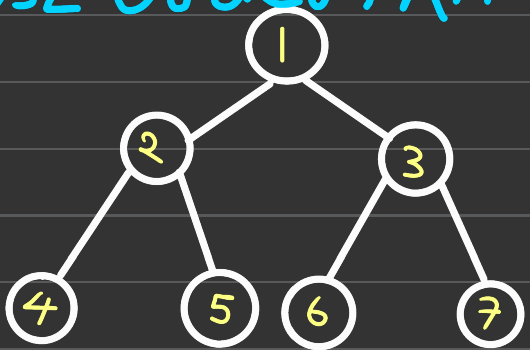
4)



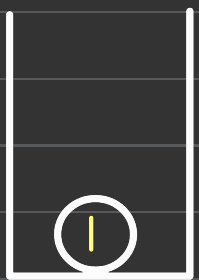
5)



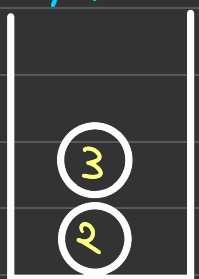
Post-Order Traversal...



Push root of the tree...



Step 1>



Steps: taken:

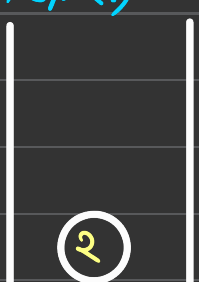
:) Push ① to the ans vector And

⇒ pop it from the stack..

:) Now Push left subtree node to the stack followed by right sub-tree node (if it exists)

⇒ Repeat while (stack.size() > 0);

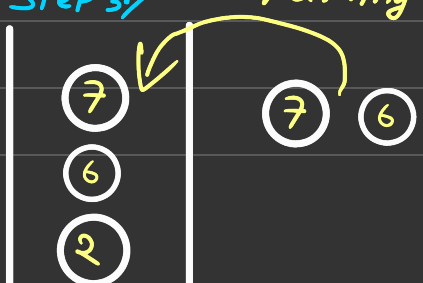
Step 2:)



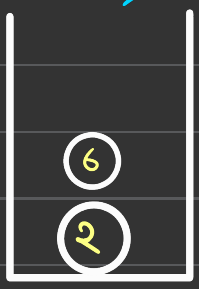
Push

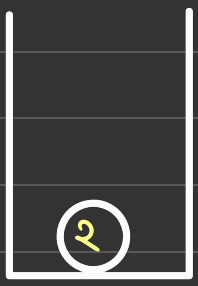
Step 3:>

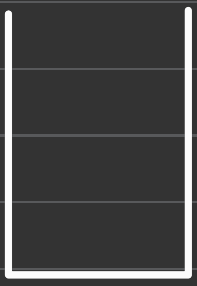
Pushing Subtree of 3)

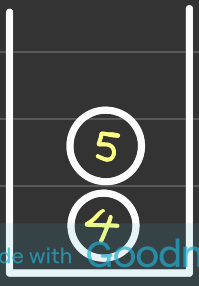


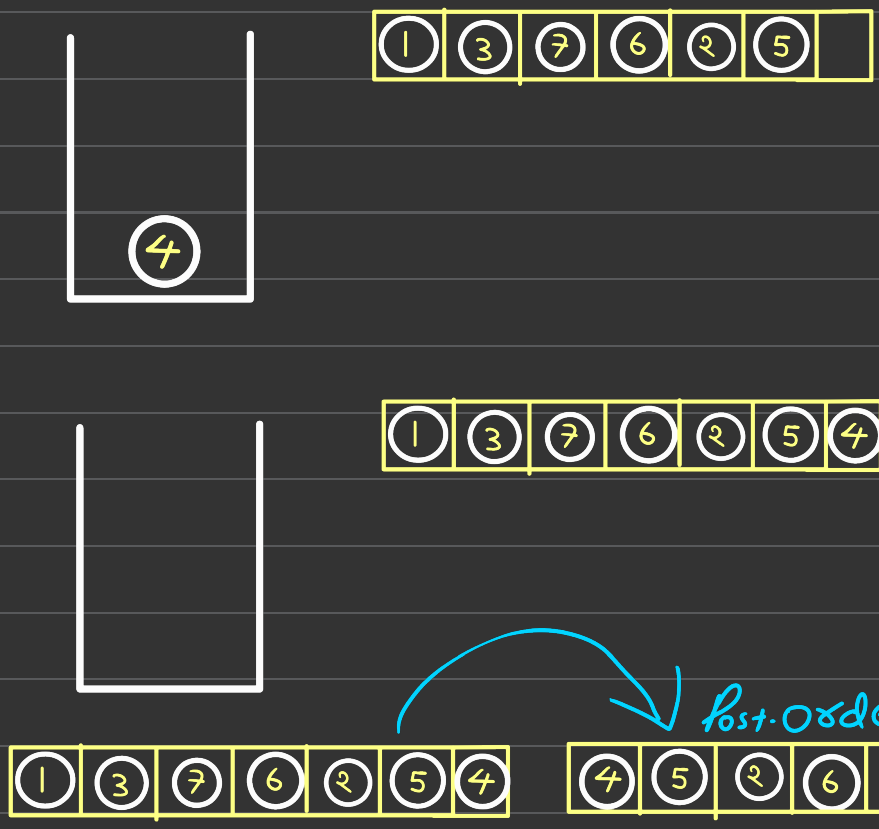
Step 4)



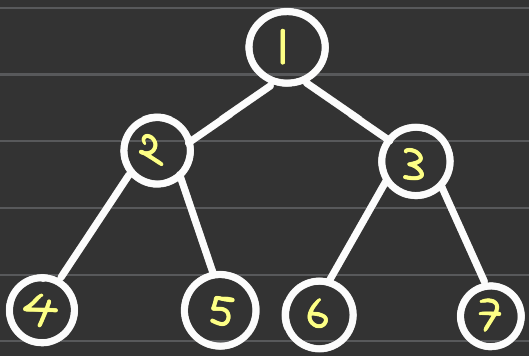


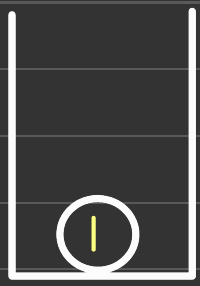


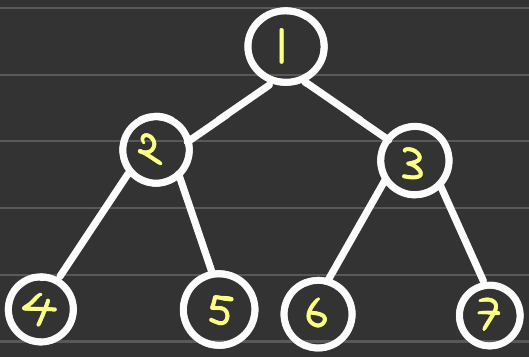


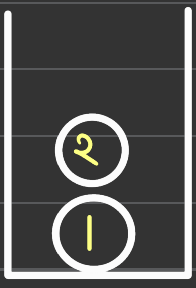


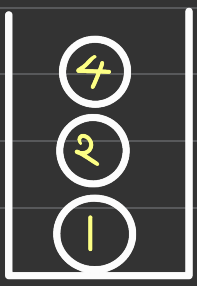
In-Order Traversal:-

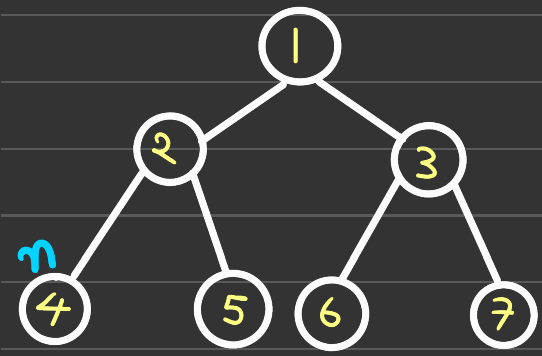


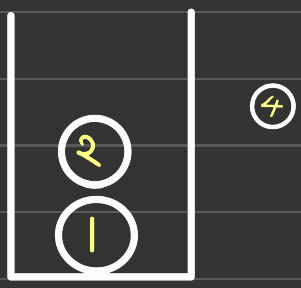




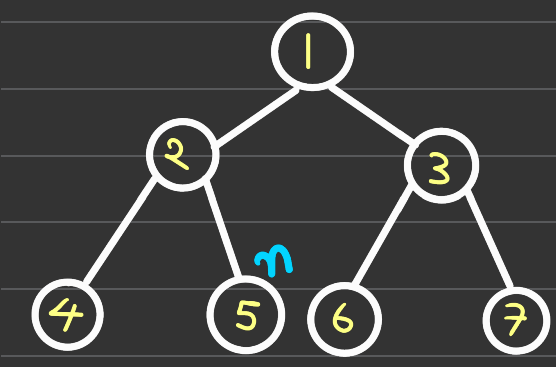


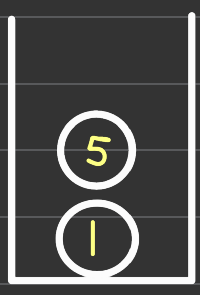




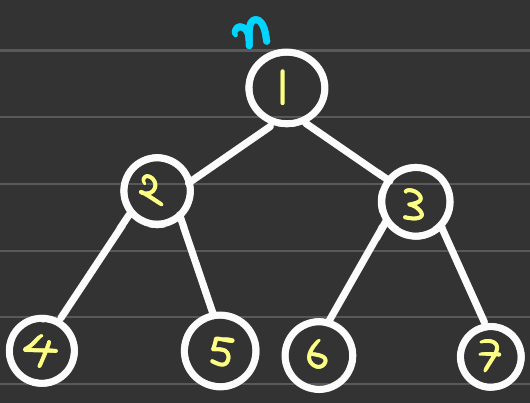


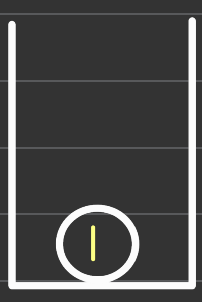
4



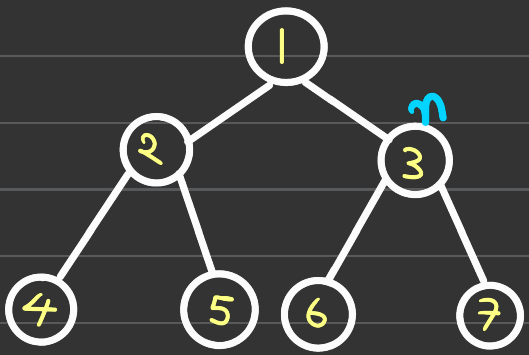


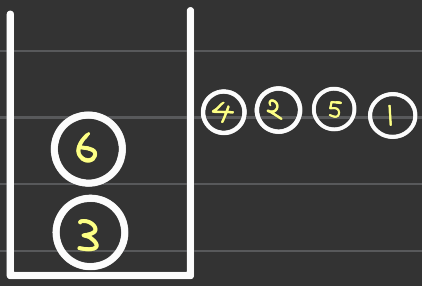
4 2

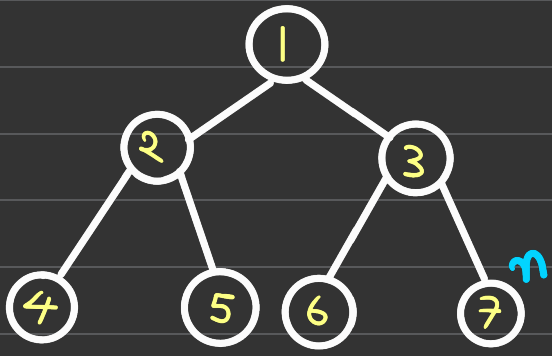


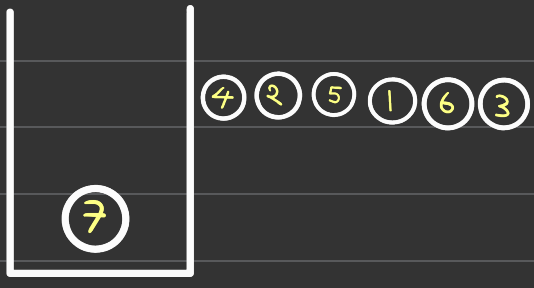


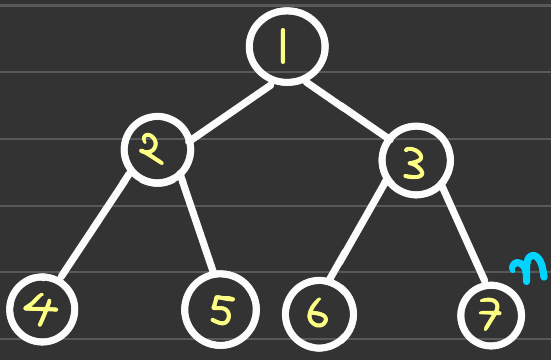
4 2 5

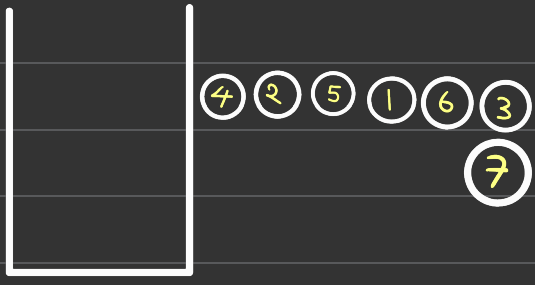




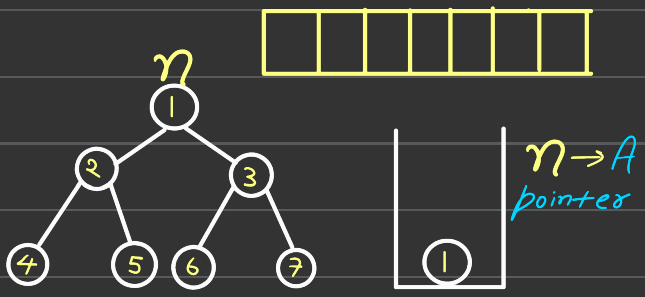


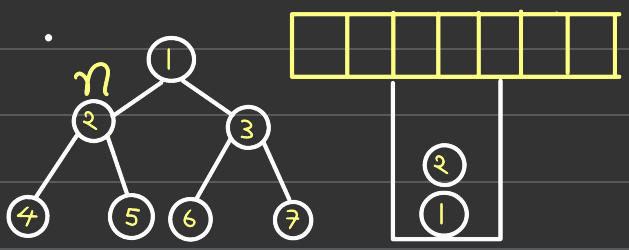


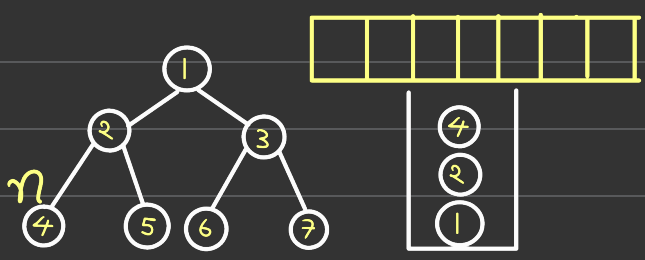


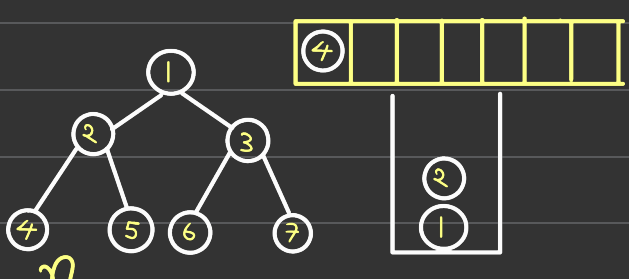


Algorithm of In-ORDER



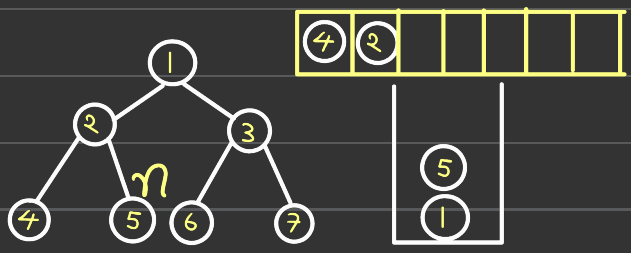


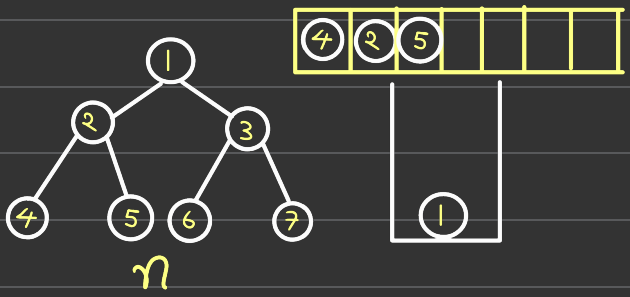


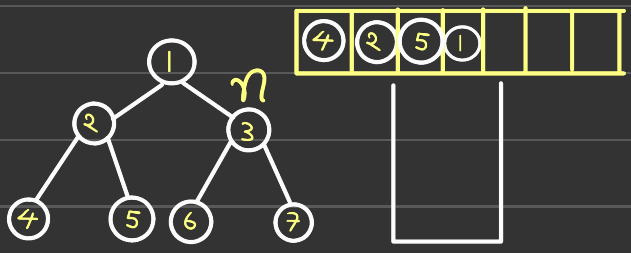


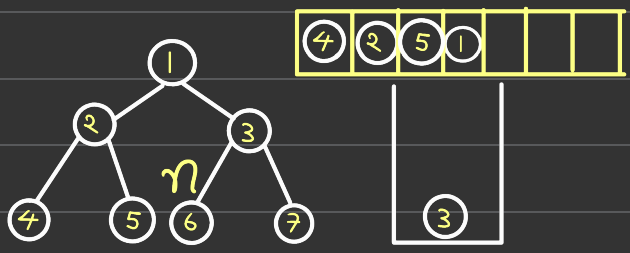
Algorithm:-

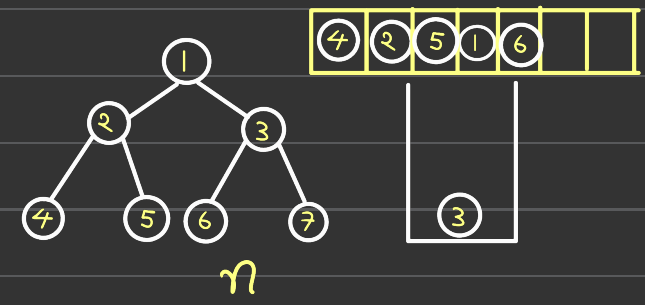
```
while (St.size() > 0){
 if (n != null){
   St.push (n->val);
   n = n->next;
 }
 else {
 node* temp = St.top();
 St. pop();
 Visit(display || push)
   n = temp->right; }
```

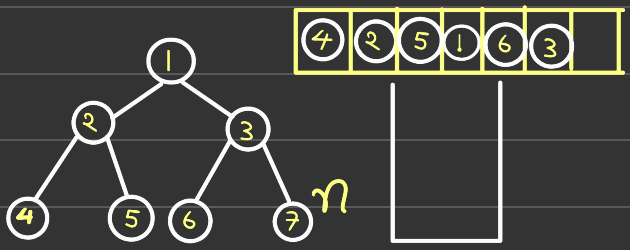


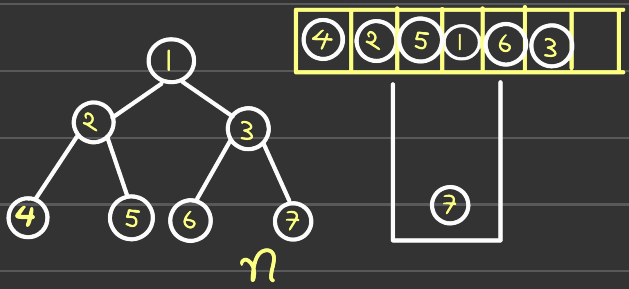


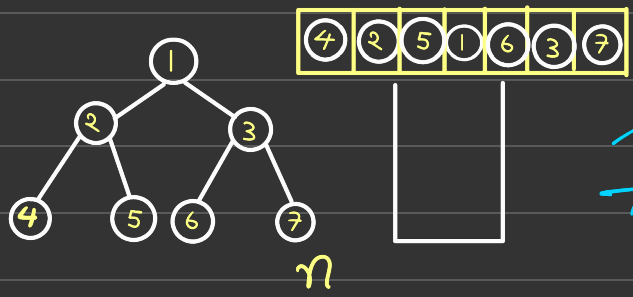












Pointed preorder traversal...

Code Snippet:—

```cpp
void pre_iterative(node* root){
  stack<node*> st;
  st.push(root);
  while(st.size()>0){
    node* temp = st.top();
    st.pop();
    cout<<temp->val<<" ";
    if(temp->right) st.push(temp->right);
    if(temp->left) st.push(temp->left);
  }
}
```

```cpp
void in_order_traversal(node* root){
  stack<node*> st;
  node* n = root;
  while(st.size()>0 || n){
    if(n!=NULL){
      st.push(n);
      n = n -> left;
    }
    else{
      node* temp = st.top();
      st.pop();
      cout<<temp->val<<" ";
      n = temp -> right;
    }
  }
}
```

```cpp
void post_iterative(node* root){
  vector<int>ans;
  stack<node*>st;
  if(root) st.push(root);
  while(st.size()>0){
    node* temp = st.top();
    st.pop();
    ans.push_back(temp->val);
    if(temp->left)st.push(temp->left);
    if(temp->right)st.push(temp->right);
  }
  reverse(ans.begin(),ans.end());
  for(int i=0;i<ans.size();i++){
    cout<<ans[i]<<" ";
  }
}
```