```
void display (struct Node * root)
{ Inorder (root);
}
```

Sneha SB
9/12/25

Code

```
#include <stdio.h>
#include <stdlib.h>
struct Node
{ int data;
  struct Node * left;
  struct Node * right;
};
struct Node * createNode (struct Node * root, int val)
{ struct Node * newNode = (struct Node *) malloc(sizeof(struct Node));
  newNode -> data = val;
  newNode -> left = newNode -> right = NULL;
  return newNode;
}
struct Node * Insert (struct Node * root, int val)
{ if (root == NULL)
      return createNode(root, val);
  if (root -> data > val)
      root -> left = Insert(root -> left, val);
  else if (root -> data < val)
      root -> right = Insert(root -> right, val);
  return root;
};
void Inorder (struct Node * root)
{ if (root == NULL)
      return;
  Inorder (root -> left);
  printf ("%d", root -> data);
  Inorder (root -> right);
}
```

```
void preorder(struct Node *root)
{ if (root==NULL)
      return;
   printf("%d ", root->data);
   preorder(root->left);
   preorder(root->right);
}
void postorder(struct Node *root)
{ if (root==NULL)
      return;
   postorder(root->left);
   postorder(root->right);
   printf("%d", root->data);
}
void display(struct Node *root)
{
   printf("----BST----\n");
   inorder(root);
   printf("\n");
}
void main()
{ struct Node *root=NULL;
  int val, ch;
  while(1){
    printf("1. Insert BST, 2. Inorder traverse, 3. preorder traverse,
        4. postorder traverse, 5. display, 6. exit \n");
    printf("enter choice:\n");
    scanf("%d", &ch);
    switch(ch)
    { case 1: printf("enter value:\n");
              scanf("%d", &val);
              root = Insert(root, val);
              break;
      case 2: inorder(root);
              break;
      case 3: preorder(root);
              break;
      case 4: postorder(root);
              break;
      case 5: display(root);
              break;
      case 6: exit(0);
      default: printf("Invalid choice\n");
    }
  }
  return 0;
}
```

Output:

1. Insert into BST, 2. Inorder traverse, 3. preorder traverse, 4. postorder traverse, 5. display, 6. exit.

enter choice:
1
enter value:
11
enter choice:
1
enter value:
7
enter choice:
1
enter value:
20
enter choice:
1
enter value:
31
enter choice:
1
enter value:
18
enter choice:
1
enter value
43
enter choice:
1
enter value:
12
enter choice:
2
7 11 12 18 20 31 43 enter choice:
3
11 7 20 18 12 31 43 enter choice:
4
7 12 18 43 31 20 11 enter choice:
5
----BST----
7 11 12 18 20 31 43
enter choice: 6.

# leetcode problem

## Merge two Binary trees

```
struct TreeNode* mergeTrees(struct TreeNode *root1, struct TreeNode *root2)
{ if (root1==NULL)
    return root2;
  if (root2==NULL)
    return root1;
  if (root1==NULL && root2==NULL)
    return NULL;
  root1->val = root1->val + root2->val;
  root1->left = mergeTrees(root1->left, root2->left);
  root2->right = mergeTrees(root1->right, root2->right);
  return root1;
}
```

output:

input:

root1 = [1,3,2,5] , root2 = [2,1,3,null,4,null,7]

Output: [3,4,5,5,4,null,7]

Both o/p same