Name- Abhinav Kumar
PRN- 21070126006
Branch- AIML-A1
Lab Assignment- Binary Search Tree

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int value;
struct node *left, *right;
};
struct node *new_node(int value)
{
struct node *tmp = (struct node *)malloc(sizeof(struct node));
tmp->value = value;
tmp->left = tmp->right = NULL;
return tmp;
}
void print(struct node *root_node)
{
if (root_node != NULL)
{
print(root_node->left);
printf("%d \n", root_node->value);
print(root_node->right);
}
}
struct node* insert_node(struct node* node, int value)
{
if (node == NULL)
return new_node(value);
if (value < node->value)
{
node->left = insert_node(node->left, value);
}
else if (value > node->value)
{
node->right = insert_node(node->right, value);
}
return node;
}

int main()
{

struct node *root_node = NULL;
root_node = insert_node(root_node, 30);
insert_node(root_node, 30);
insert_node(root_node, 29);
insert_node(root_node, 45);
```

```
insert_node(root_node, 55);
insert_node(root_node, 76);
insert_node(root_node, 85);

print(root_node);

return 0;
}
 Binary Tree in C++

#include <stdlib.h>
#include <iostream>

using namespace std;
struct node {
  int data;
  struct node *left;
  struct node *right;
};
struct node *newNode(int data) {
  struct node *node = (struct node *)malloc(sizeof(struct node));
  node->data = data;
  node->left = NULL;
  node->right = NULL;
  return (node);
}
void traversePreOrder(struct node *temp) {
  if (temp != NULL) {
    cout << " " << temp->data;
    traversePreOrder(temp->left);
    traversePreOrder(temp->right);
  }
}
void traverseInOrder(struct node *temp) {
  if (temp != NULL) {
    traverseInOrder(temp->left);
    cout << " " << temp->data;
    traverseInOrder(temp->right);
  }
}
void traversePostOrder(struct node *temp) {
  if (temp != NULL) {
    traversePostOrder(temp->left);
    traversePostOrder(temp->right);
    cout << " " << temp->data;
  }
}
int main() {
  struct node *root = newNode(1);
  root->left = newNode(2);
```

```
  root->right = newNode(3);
  root->left->left = newNode(4);
  cout << "preorder traversal: ";
  traversePreOrder(root);
  cout << "\nInorder traversal: ";
  traverseInOrder(root);
  cout << "\nPostorder traversal: ";
  traversePostOrder(root);
}
```