Page No.:
Date:

Experiment no-7

Aim - Implement binary tree using linked list & perform recursive traversals.

Theory -

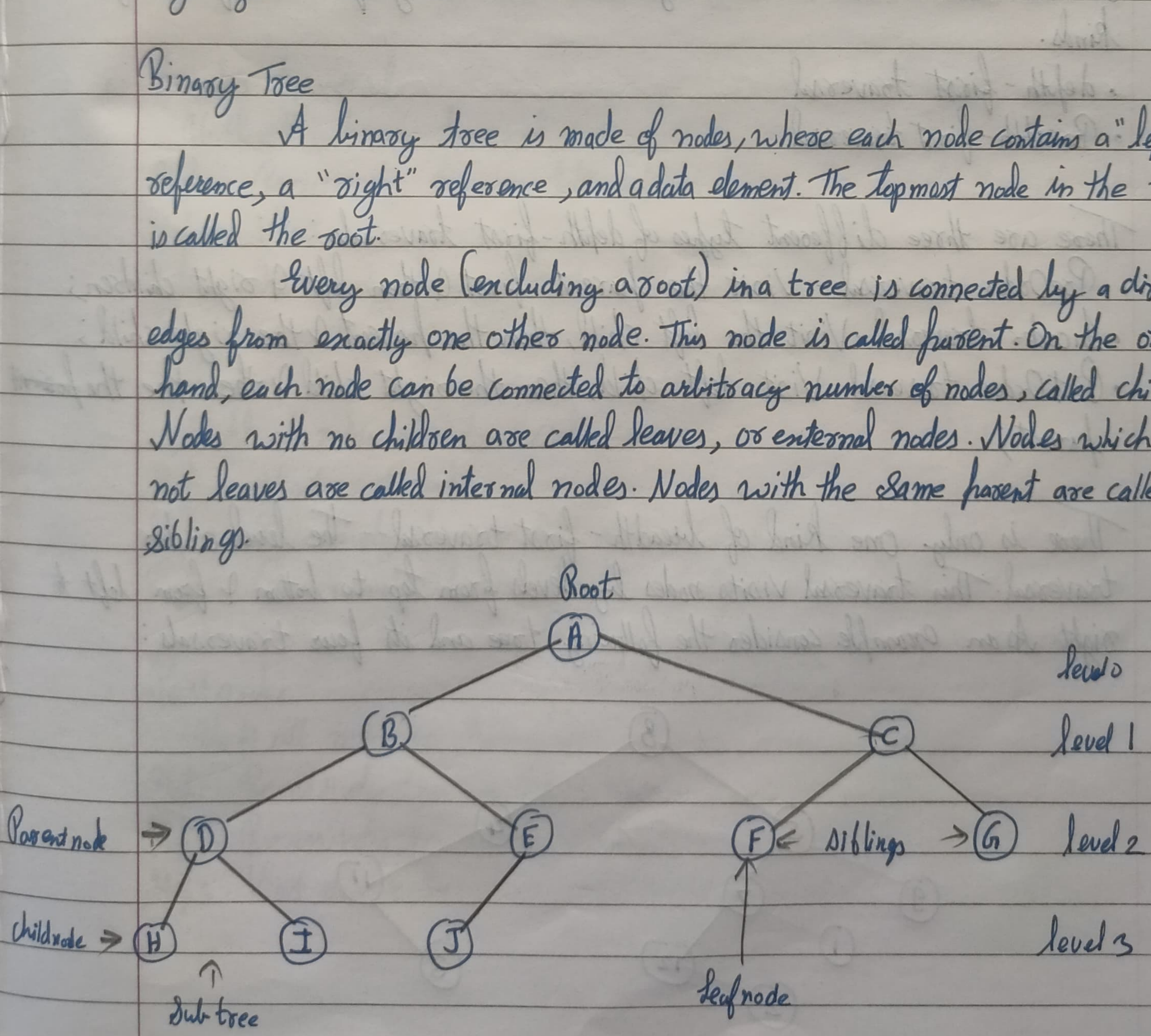
Tree

Tree represents the nodes connected by edges also a class of graphs that is acyclic is termed as trees. Let us now discuss an important class of graphs called trees and its associated terminology.
Trees are useful in describing any structure that involves hierarchy. Familiar examples of such structures are family trees, the hierarchy of positions in an organization, and so on.

Binary Tree

A binary tree is made of nodes, where each node contains a "left" reference, a "right" reference, and a data element. The topmost node in the tree is called the root.

Every node (excluding a root) in a tree is connected by a directed edges from exactly one other node. This node is called parent. On the other hand, each node can be connected to arbitrary number of nodes, called children. Nodes with no children are called leaves, or external nodes. Nodes which are not leaves are called internal nodes. Nodes with the same parent are called siblings.



Page No.:
Date:

Insert Operation

The very first insertion creates the trees. Afterwards, whenever an element is to be inserted first locate its proper location. Start searching from the root nodes, then if the data is less than the key value, search for the empty location in the left subtree & insert the data. Otherwise, search for the empty location in the right subtree & insert the data.

Traversals

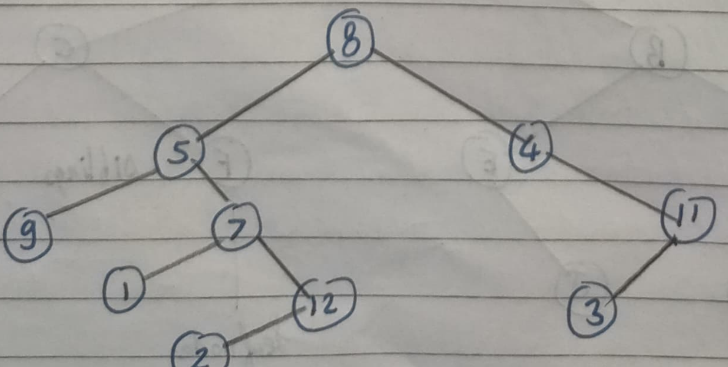
A traversal is a process that visits all the nodes in the tree. Since a tree is a nonlinear data structure, there is no unique traversal. We will consider ~~[illegible]~~ several traversal algorithms with we graph in the following two kinds.

- depth-first traversal
- Breadth-first traversal

There are three different types of depth-first traversals:

- PreOrder traversal - Visit the parent first & then left & right children;
- Inorder traversal - Visit the left child, then the parent & the right child;
- PostOrder traversal - Visit left child, then the right child & then the the ~~parent~~ parent;

There is only one kind of breadth-first traversal -- the level order traversal. This traversal visits nodes by levels from top to bottom & from left to right. As an example consider the following tree and its four traversals:



Page No.:
Date:

Algorithm:

Algorithm to insert a node:

Step 1 - Search for the node whose child node is to be inserted. This is a node at some level i, & a node is to be inserted at the level i+1 as either its left child or right child. This is the node after which the insertion is to be made.

Step 2 - Link a new node to the nodes that becomes its parent node, that is, either the Lchild or the Rchild.

Algorithm to traverse a tree:

- Inorder traversal
  Until all nodes are traversed -
  Step 1 - Recursively traverse left subtree.
  Step 2 - Visit root node
  Step 3 - Recursively traverse right subtree.

- Preorder
  Until all nodes are traversed
  Step 1 - Visit root node.
  Step 2 - Recursively traverse left subtree.
  Step 3 - Recursively traverse right subtree.

- Postorder
  Until all nodes are traversed -
  Step 1 - Recursively traverse left subtree.
  Step 2 - Recursively traverse right subtree
  Step 3 - Visit root node

Page No.:

Date:

Algorithm to copy one tree into another tree:

```
Step 1- If (Root = = Null)
Then return Null
Step 2- Tmp = New TreeNode
Step 3- Tmp -> Lchild = TreeCopy (Root -> Lchild);
Step 4- Tmp -> Rchild = TreeCopy (Root -> Rchild);
Step 5- Tmp-Data = Then return
```

Outcome:

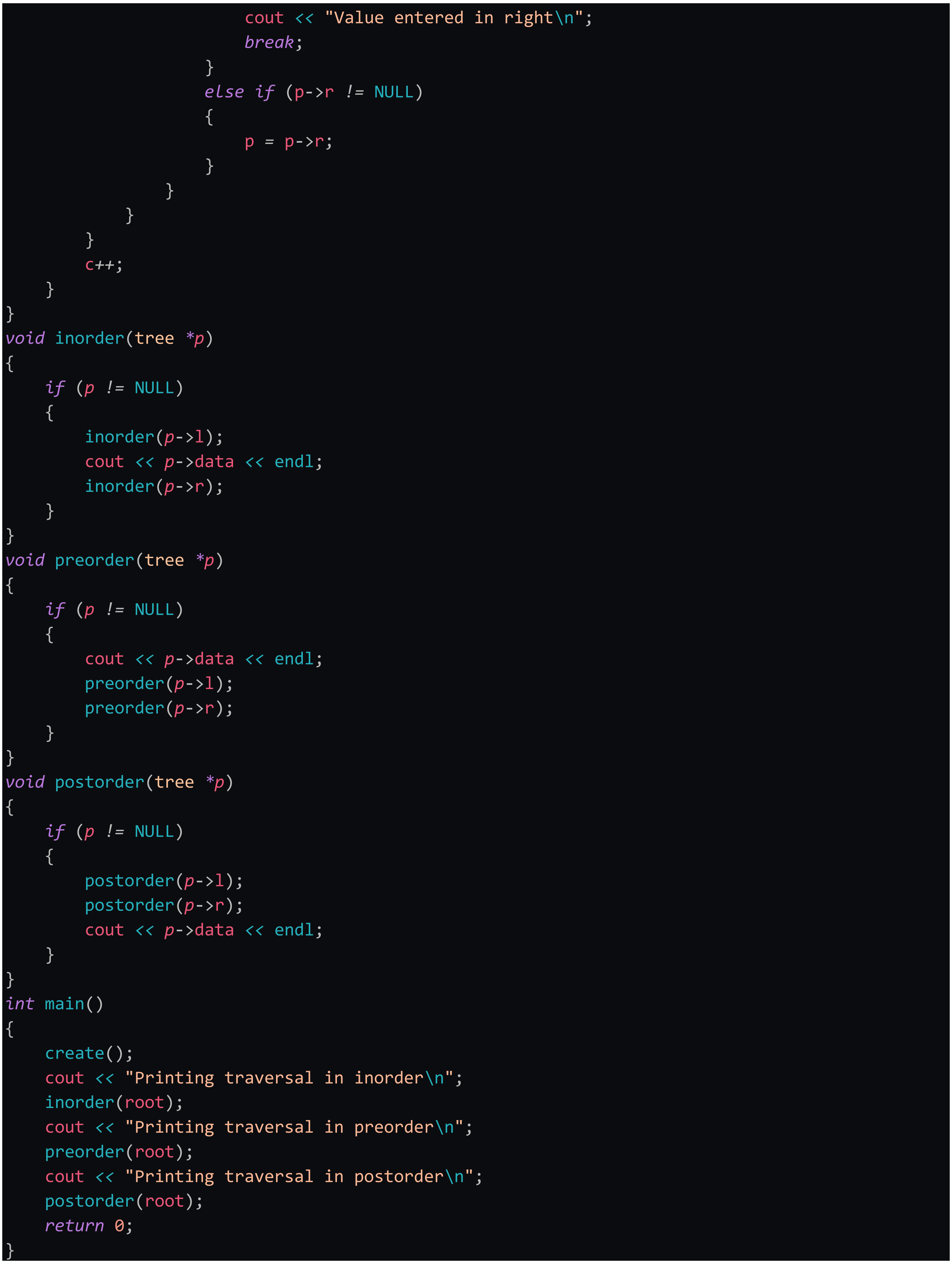
Learn Object oriented programming features.

Understand & implement different operations on tree & binary tree.

Conclusion: Thus we have studied the implementation of various Binary tree operations.

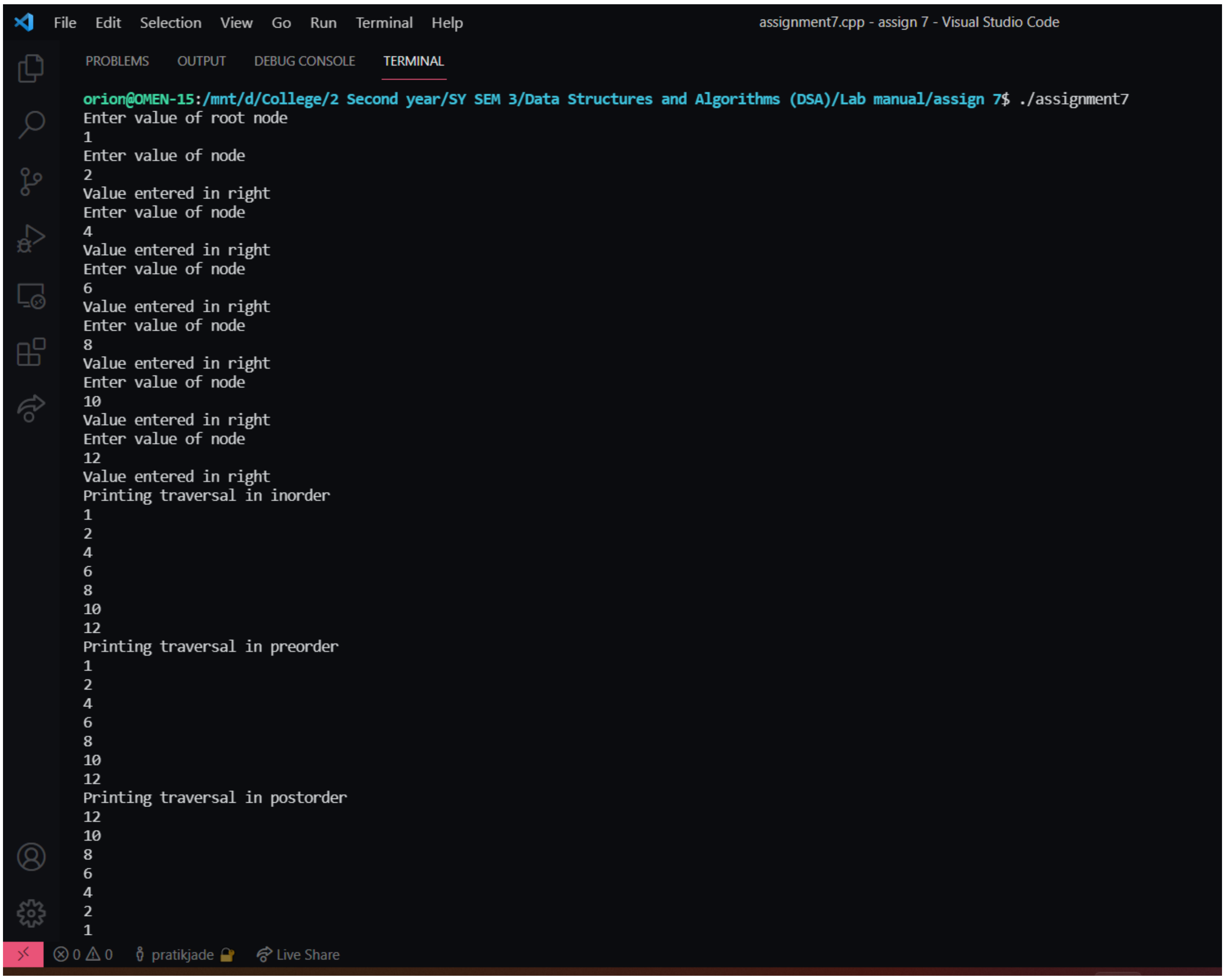
Program –

```
#include <iostream>
using namespace std;
struct tree
{
    tree *l, *r;
    int data;
} *root = NULL, *p = NULL, *np = NULL, *q;
void create()
{
    int value, c = 0;
    while (c < 7)
    {
        if (root == NULL)
        {
            root = new tree;
            cout << "Enter value of root node\n";
            cin >> root->data;
            root->r = NULL;
            root->l = NULL;
        }
        else
        {
            p = root;
            cout << "Enter value of node\n";
            cin >> value;
            while (true)
            {
                if (value < p->data)
                {
                    if (p->l == NULL)
                    {
                        p->l = new tree;
                        p = p->l;
                        p->data = value;
                        p->l = NULL;
                        p->r = NULL;
                        cout << "Value entered in left\n";
                        break;
                    }
                    else if (p->l != NULL)
                    {
                        p = p->l;
                    }
                }
                else if (value > p->data)
                {
                    if (p->r == NULL)
                    {
                        p->r = new tree;
                        p = p->r;
                        p->data = value;
                        p->l = NULL;
                        p->r = NULL;
```



```
                        cout << "Value entered in right\n";
                        break;
                    }
                    else if (p->r != NULL)
                    {
                        p = p->r;
                    }
                }
            }
        }
        c++;
    }
}
void inorder(tree *p)
{
    if (p != NULL)
    {
        inorder(p->l);
        cout << p->data << endl;
        inorder(p->r);
    }
}
void preorder(tree *p)
{
    if (p != NULL)
    {
        cout << p->data << endl;
        preorder(p->l);
        preorder(p->r);
    }
}
void postorder(tree *p)
{
    if (p != NULL)
    {
        postorder(p->l);
        postorder(p->r);
        cout << p->data << endl;
    }
}
int main()
{
    create();
    cout << "Printing traversal in inorder\n";
    inorder(root);
    cout << "Printing traversal in preorder\n";
    preorder(root);
    cout << "Printing traversal in postorder\n";
    postorder(root);
    return 0;
}
```

Output-



```
orion@OMEN-15:/mnt/d/College/2 Second year/SY SEM 3/Data Structures and Algorithms (DSA)/Lab manual/assign 7$ ./assignment7
Enter value of root node
1
Enter value of node
2
Value entered in right
Enter value of node
4
Value entered in right
Enter value of node
6
Value entered in right
Enter value of node
8
Value entered in right
Enter value of node
10
Value entered in right
Enter value of node
12
Value entered in right
Printing traversal in inorder
1
2
4
6
8
10
12
Printing traversal in preorder
1
2
4
6
8
10
12
Printing traversal in postorder
12
10
8
6
4
2
1
```

A72 Pratik Jade