Experiment 6

Student Name: Harsh Kumar **UID: 22BCS15754**
Branch: BE-CSE **Section/Group: 603_FL_IOT-B**
Semester: 5th **Date of Performance:26/08/2024**
Subject Name: AP **Subject Code: 22CSP-314**

1. **Aim:** Write a program to print the elements in inorder, preorder, and postorder traversal of the Binary Search Tree.

2. **Objective:** The objective of this program is to print the elements in inorder, preorder, and postorder traversal of the Binary Search Tree.

3. **Implementation/Code:**

```
#include <iostream>
using namespace std;

class Node {
public:
    int data;
    Node* left;
    Node* right;
    Node(int v)
    {
        this->data = v;
        this->left = this->right = NULL;
    }
};

// Inorder Traversal
void printInorder(Node* node)
{
    if (node == NULL)
        return;
    // Traverse left subtree
    printInorder(node->left);
```

```
    // Visit node
    cout << node->data << " ";
    // Traverse right subtree
    printInorder(node->right);
}

// Preorder Traversal
void printPreOrder(Node* node)
{
    if (node == NULL)
        return;
    // Visit Node
    cout << node->data << " ";
    // Traverse left subtree
    printPreOrder(node->left);
    // Traverse right subtree
    printPreOrder(node->right);
}

// PostOrder Traversal
void printPostOrder(Node* node)
{
    if (node == NULL)
        return;
    // Traverse left subtree
    printPostOrder(node->left);
    // Traverse right subtree
    printPostOrder(node->right);
    // Visit node
    cout << node->data << " ";
}

int main()
{
    // Build the tree
    Node* root = new Node(100);
    root->left = new Node(20);
    root->right = new Node(200);
    root->left->left = new Node(10);
    root->left->right = new Node(30);
    root->right->left = new Node(150);
    root->right->right = new Node(300);
```

```
    // Function call
    cout << "Inorder Traversal: ";
    printInorder(root);
    cout<<"\n";
    cout << "Preorder Traversal: ";
    printPreOrder(root);
    cout<<"\n";
    cout << "PostOrder Traversal: ";
    printPostOrder(root);
    return 0;
}
```

4. Output:

```
Inorder Traversal: 10 20 30 100 150 200 300
Preorder Traversal: 100 20 10 30 200 150 300
PostOrder Traversal: 10 30 20 150 300 200 100

...Program finished with exit code 0
Press ENTER to exit console.
```

1.(ii) Aim: Write a program to print the top view of the given binary tree.

2.(ii) Objective: The objective of a program to print the nodes in the top view of a binary tree is to provide a view of the tree from the top, essentially showing which nodes are visible when the tree is viewed from above.

3.(ii) Implementation/Code:

```
#include <bits/stdc++.h>
using namespace std;

// Structure of binary tree
struct Node {
    Node* left;
    Node* right;
    int hd;
```

```
    int data;
};
// function to create a new node
Node* newNode(int key)
{
    Node* node = new Node();
    node->left = node->right = NULL;
    node->data = key;
    return node;
}
// function should print the topView of
// the binary tree
void topview(Node* root)
{
    if (root == NULL)
            return;
    queue<Node*> q;
    map<int, int> m;
    int hd = 0;
    root->hd = hd;
    // push node and horizontal distance to queue
    q.push(root);
    cout << "The top view of the tree is : \n";
    while (q.size()) {
            hd = root->hd;
            // count function returns 1 if the container
            // contains an element whose key is equivalent
            // to hd, or returns zero otherwise.
            if (m.count(hd) == 0)
                    m[hd] = root->data;
            if (root->left) {
                    root->left->hd = hd - 1;
                    q.push(root->left);
            }
            if (root->right) {
                    root->right->hd = hd + 1;
                    q.push(root->right);
            }
```

```
            q.pop();
            root = q.front();
        }
        for (auto i = m.begin(); i != m.end(); i++) {
            cout << i->second << " ";
        }
    }
    // Driver code
    int main()
    {
        Node* root = newNode(1);
        root->left = newNode(2);
        root->right = newNode(3);
        root->left->right = newNode(4);
        root->left->right->right = newNode(5);
        root->left->right->right->right = newNode(6);
        topview(root);
        return 0;
    }
```

4.(ii)Output:

```
The top view of the tree is :
2 1 3 6

...Program finished with exit code 0
Press ENTER to exit console.
```