```cpp
// Question1.cpp : This file contains the 'main' function. Program execution begins and ends there.
////////// / -----------------------------------------------------------------------------------------------------------------------
//Name                              Sai Chaitanya Kilambi
//Course                            CPSC 131 Data Structures, Fall, 2022
//Assignment                        No.11 question:1
//Due date                          11/30/2022
// Purpose:
// This program demonstrates insertion of data in array Days into a Binary Search Tree along with preorder,inorder
// and postorder traversal. It also demonstrates how to search display and count the number of leaves
//------------------------------------------------------------------------------------------------------------------------
// list of libraries
//
//importing the required libraries

#include <iostream>
using namespace std;

//Node class
class Node {
public:
    string data;
    Node* left;
    Node* right;
    //constructor
    Node(string data) {
        this->data = data;
        left = NULL;
        right = NULL;
    }
};

//BST class modified
class BST
{
    //root of the tree as attribute.
private:
    Node* root;


public:
    //constructors
    BST() {
        root = NULL;
    }
    BST(string data) {
        root = new Node(data);
    }


    //one private helper method for each utility function.
private:
    //insert helper function
    void insert(Node*& root, string data) {
        if (root == NULL) {
            root = new Node(data);
        }
        else
        {
            if (root->data > data) {
                insert(root->left, data);
            }
            else {
                insert(root->right, data);
            }
        }
    }
public:
    //insert function
    void insert(string data) {
        insert(root, data);
    }


private:
    //inorder helper function
    void inorder(Node* curr) {
        if (curr == NULL) {
            return;
        }

        inorder(curr->left);
        cout << curr->data << " ";
        inorder(curr->right);
    }
public:
    //inorder function
    void inorder() {
        cout << "Inorder traversal is :- ";
        inorder(root);
        cout << endl;
    }

```

```

private:
    //postorder helper function
    void postorder(Node* curr) {
        if (curr == NULL) {
            return;
        }

        inorder(curr->left);
        inorder(curr->right);
        cout << curr->data << " ";
    }
public:
    //postorder function
    void postorder() {
        cout << "Postorder traversal is:- ";
        postorder(root);
        cout << endl;
    }


private:
    //preorder helper function
    void preorder(Node* curr) {
        if (curr == NULL) {
            return;
        }

        cout << curr->data << " ";
        inorder(curr->left);
        inorder(curr->right);
    }
public:
    //preorder function
    void preorder() {
        cout << "Preorder traversal is:- ";
        preorder(root);
        cout << endl;
    }


private:
    //display leaves helper function
    void displayleaves(Node* curr) {
        if (curr == NULL) {
            return;
        }
        if (curr->left == NULL && curr->right == NULL) {
            cout << curr->data << " ";
```

```cpp
            return;
        }
        displayleaves(curr->left);
        displayleaves(curr->right);
    }
public:
    //display leaves function
    void displayleaves() {
        cout << "Only leaves of tree are:- ";
        displayleaves(root);
        cout << endl;
    }


private:
    //display nodes with one child helper function
    void Nodewithonechild(Node* curr) {
        if (curr == NULL)
            return;
        if ((curr->left == NULL) ^ (curr->right == NULL)) {
            cout << curr->data << " ";
        }
        Nodewithonechild(curr->left);
        Nodewithonechild(curr->right);
    }
public:
    //display nodes with one child function
    void Nodewithonechild() {
        cout << "Nodes with one child are:- ";
        Nodewithonechild(root);
        cout << endl;
    }


private:
    //height helper function
    int height(Node* curr) {
        if (curr == 0)
            return 0;
        return 1 + max(height(curr->left), height(curr->right));
    }
public:
    //height function
    int height() {
        return height(root);
    }

private:
    //search helper function
    void search(Node* curr, string data) {
        if (curr == NULL) {
            cout << data << " not found" << endl;
            return;
        }

        if (curr->data == data) {
            cout << data << " found" << endl;
            return;
        }

        if (curr->data > data) {
            search(curr->left, data);
        }
        else {
            search(curr->right, data);
        }
    }
public:
    //search function
    void search(string data) {
        cout << "Searching " << data << ":- ";
        search(root, data);
    }

private:
    //number of nodes helper function
    int NumberofNodes(Node* curr) {
        if (curr == 0)
            return 0;
        return 1 + NumberofNodes(curr->left) + NumberofNodes(curr->right);
    }
public:
    //number of nodes function
    int NumberofNodes() {
        return NumberofNodes(root);
    }
};

//main
int main() {
    //given array
    string Days[7] = { "MON","TUE","WED","THR","FRI","SAT","SUN" };

    BST* tree = new BST();
    for (string day : Days)
        tree->insert(day);

    tree->inorder();
    tree->postorder();
    tree->preorder();
    tree->displayleaves();
    tree->Nodewithonechild();
    int height = tree->height();
    cout << "Height of BST is " << height << endl;
    //searching for Mon
    tree->search("Mon");
    //searching for THR
    tree->search("THR");
    cout << "Number of nodes in BST is " << tree->NumberofNodes() << endl;
}
```