Assignment 4

Problem Statement:-

Beginning with an empty binary search tree, Construct binary search tree by inserting the values in the order given. After constructing a binary tree -
i. Insert new node, ii. Find number of nodes in longest path from root, iii. Minimum data value found in the tree, iv. Change a tree so that the roles of the left and right pointers are swapped at every node, v. Search a value.

Source Code:-

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

// Structure for a Node
struct Node {
    int data;
    Node* left;
    Node* right;

    Node(int value) {
        data = value;
        left = right = nullptr;
    }
};

// Class for Binary Search Tree
class BinarySearchTree {
public:
    Node* root;

    BinarySearchTree() {
        root = nullptr;
    }

    // Insert a node into the BST
```

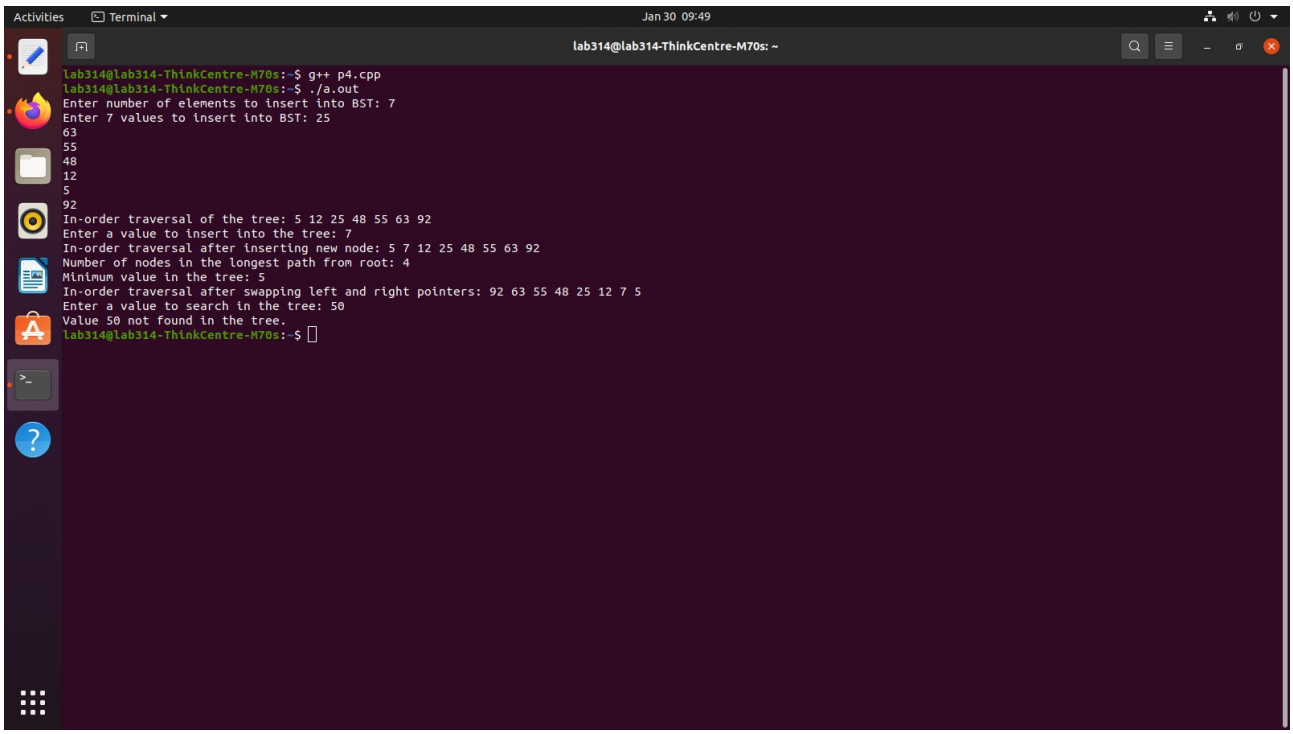
```
void insert(int value) {
    root = insertHelper(root, value);
}

// Function to insert a node into the BST
Node* insertHelper(Node* node, int value) {
    if (node == nullptr) {
        return new Node(value);
    }
    if (value < node->data) {
        node->left = insertHelper(node->left, value);
    } else {
        node->right = insertHelper(node->right, value);
    }
    return node;
}

// Find the number of nodes in the longest path from root
int longestPath() {
    return longestPathHelper(root);
}

int longestPathHelper(Node* node) {
    if (node == nullptr) {
        return 0;
    }
    int leftDepth = longestPathHelper(node->left);
    int rightDepth = longestPathHelper(node->right);
    return max(leftDepth, rightDepth) + 1;
}

// Find the minimum value in the BST
int findMinValue() {
    if (root == nullptr) {
        cout << "Tree is empty!" << endl;
        return -1;
    }
    return findMinValueHelper(root);
```

```
}

int findMinValueHelper(Node* node) {
    while (node->left != nullptr) {
        node = node->left;
    }
    return node->data;
}

// Swap left and right pointers at every node
void swapChildren() {
    swapChildrenHelper(root);
}

void swapChildrenHelper(Node* node) {
    if (node == nullptr) {
        return;
    }
    swap(node->left, node->right);
    swapChildrenHelper(node->left);
    swapChildrenHelper(node->right);
}

// Search for a value in the BST
bool search(int value) {
    return searchHelper(root, value);
}

bool searchHelper(Node* node, int value) {
    if (node == nullptr) {
        return false;
    }
    if (node->data == value) {
        return true;
    }
    if (value < node->data) {
        return searchHelper(node->left, value);
    } else {
        return searchHelper(node->right, value);
```

```
        }
    }

    // Function to print the tree in-order (for visualization)
    void inorder() {
        inorderHelper(root);
        cout << endl;
    }

    void inorderHelper(Node* node) {
        if (node != nullptr) {
            inorderHelper(node->left);
            cout << node->data << " ";
            inorderHelper(node->right);
        }
    }
};

int main() {
    BinarySearchTree tree;
    int n, value;

    cout << "Enter number of elements to insert into BST: ";
    cin >> n;

    cout << "Enter " << n << " values to insert into BST: ";
    for (int i = 0; i < n; ++i) {
        cin >> value;
        tree.insert(value);
    }

    cout << "In-order traversal of the tree: ";
    tree.inorder();

    // i. Insert a new node
    cout << "Enter a value to insert into the tree: ";
    cin >> value;
    tree.insert(value);
    cout << "In-order traversal after inserting new node: ";
```

```
    tree.inorder();

    // ii. Find the number of nodes in the longest path from root
    cout << "Number of nodes in the longest path from root: " <<
tree.longestPath() << endl;

    // iii. Minimum data value found in the tree
    cout << "Minimum value in the tree: " << tree.findMinValue() << endl;

    // iv. Change the tree so that the roles of the left and right pointers are
swapped at every node
    tree.swapChildren();
    cout << "In-order traversal after swapping left and right pointers: ";
    tree.inorder();

    // v. Search for a specific value
    cout << "Enter a value to search in the tree: ";
    cin >> value;
    if (tree.search(value)) {
        cout << "Value " << value << " found in the tree." << endl;
    } else {
        cout << "Value " << value << " not found in the tree." << endl;
    }

    return 0;
}
```

Output:-


Activities
Terminal
Jan 30 09:49
lab314@lab314-ThinkCentre-M70s: ~
lab314@lab314-ThinkCentre-M70s:~$ g++ p4.cpp
lab314@lab314-ThinkCentre-M70s:~$ ./a.out
Enter number of elements to insert into BST: 7
Enter 7 values to insert into BST: 25
63
55
48
12
5
92
In-order traversal of the tree: 5 12 25 48 55 63 92
Enter a value to insert into the tree: 7
In-order traversal after inserting new node: 5 7 12 25 48 55 63 92
Number of nodes in the longest path from root: 4
Minimum value in the tree: 5
In-order traversal after swapping left and right pointers: 92 63 55 48 25 12 7 5
Enter a value to search in the tree: 50
Value 50 not found in the tree.
lab314@lab314-ThinkCentre-M70s:~$