

# Assignment 5

## Code:

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

struct Bstnode {
    int data;
    Bstnode* left = NULL;
    Bstnode* right = NULL;
};

class Btree {
public:
    Bstnode* root;

    Btree() {
        root = NULL;
    }

    // Function to create a new node
    Bstnode* GetNewNode(int in_data) {
        Bstnode* ptr = new Bstnode();
        ptr->data = in_data;
        return ptr;
    }

    // Insert a node into the tree
    Bstnode* insert(Bstnode* temp, int in_data) {
        if (temp == NULL) {
            return GetNewNode(in_data);
        }
        if (in_data < temp->data) {
            temp->left = insert(temp->left, in_data);
        } else {
            temp->right = insert(temp->right, in_data);
        }
        return temp;
    }

    void addNode() {
        int value;
        cout << "Enter value to insert into the tree: ";
        cin >> value;
        root = insert(root, value);
        cout << "Node " << value << " inserted successfully!" << endl;
    }

    // Find the depth of the tree (longest path from root)
    int findDepth(Bstnode* temp) {
        if (temp == NULL)
            return 0;
        return max(findDepth(temp->left), findDepth(temp->right)) + 1;
    }

    // Find the minimum value in the tree
    void findMinValue() {
```

```

    if (root == NULL) {
        cout << "The tree is empty!" << endl;
        return;
    }
    Bstnode* temp = root;
    while (temp->left != NULL) {
        temp = temp->left;
    }
    cout << "Minimum value in the tree: " << temp->data << endl;
}

// Mirror the tree (swap left and right pointers)
void mirrorTree(Bstnode* temp) {
    if (temp == NULL)
        return;
    swap(temp->left, temp->right);
    mirrorTree(temp->left);
    mirrorTree(temp->right);
}

void mirror() {
    if (root == NULL) {
        cout << "The tree is empty!" << endl;
        return;
    }
    mirrorTree(root);
    cout << "Tree mirrored successfully!" << endl;
}

// Search for a value in the tree
bool search(Bstnode* temp, int in_data) {
    if (temp == NULL)
        return false;
    if (temp->data == in_data)
        return true;
    if (in_data < temp->data)
        return search(temp->left, in_data);
    return search(temp->right, in_data);
}

void searchValue() {
    int value;
    cout << "Enter value to search: ";
    cin >> value;
    if (search(root, value)) {
        cout << "Value " << value << " found in the tree." << endl;
    } else {
        cout << "Value " << value << " not found in the tree." << endl;
    }
}

// Inorder traversal
void inorder(Bstnode* temp) {
    if (temp == NULL)
        return;
    inorder(temp->left);
    cout << temp->data << " ";
}

```

```

    inorder(temp->right);
}

void display() {
    if (root == NULL) {
        cout << "The tree is empty!" << endl;
        return;
    }
    cout << "Inorder traversal of the tree: ";
    inorder(root);
    cout << endl;
}
};

int main() {
    Btree tree;
    int choice;

    while (true) {
        cout << "\nMenu:\n"
            << "1. Insert new node\n"
            << "2. Find number of nodes in the longest path (depth)\n"
            << "3. Find minimum data value in the tree\n"
            << "4. Mirror the tree\n"
            << "5. Search for a value\n"
            << "6. Display tree\n"
            << "7. Exit\n"
            << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                tree.addNode();
                break;
            case 2:
                cout << "Number of nodes in the longest path (depth): " << tree.findDepth(tree.root) << endl;
                break;
            case 3:
                tree.findMinValue();
                break;
            case 4:
                tree.mirror();
                break;
            case 5:
                tree.searchValue();
                break;
            case 6:
                tree.display();
                break;
            case 7:
                cout << "Exiting program!" << endl;
                return 0;
            default:
                cout << "Invalid choice. Please try again!" << endl;
        }
    }
}

return 0;
}

```

## Output:

```
Activities Terminal Feb 12 15:52 student@TAEComp-01: ~/Desktop/rameshwari DSA
student@TAEComp-01:~/Desktop/rameshwari DSA$ g++ assign5.cpp
student@TAEComp-01:~/Desktop/rameshwari DSA$ ./a.out

Menu:
1. Insert new node
2. Find number of nodes in the longest path (depth)
3. Find minimum data value in the tree
4. Mirror the tree
5. Search for a value
6. Display tree
7. Exit
Enter your choice: 1
Enter value to insert into the tree: 1
Node 1 inserted successfully!

Menu:
1. Insert new node
2. Find number of nodes in the longest path (depth)
3. Find minimum data value in the tree
4. Mirror the tree
5. Search for a value
6. Display tree
7. Exit
Enter your choice: 1
Enter value to insert into the tree: 4
Node 4 inserted successfully!

Menu:
1. Insert new node
2. Find number of nodes in the longest path (depth)
3. Find minimum data value in the tree
4. Mirror the tree
5. Search for a value
6. Display tree
7. Exit
Enter your choice: 1
Enter value to insert into the tree: 7
Node 7 inserted successfully!

Menu:
1. Insert new node
2. Find number of nodes in the longest path (depth)
3. Find minimum data value in the tree
4. Mirror the tree
5. Search for a value
6. Display tree
```

```
Activities Terminal Feb 12 15:53 student@TAEComp-01: ~/Desktop/rameshwari DSA
4. Mirror the tree
5. Search for a value
6. Display tree
7. Exit
Enter your choice: 1
Enter value to insert into the tree: 7
Node 7 inserted successfully!

Menu:
1. Insert new node
2. Find number of nodes in the longest path (depth)
3. Find minimum data value in the tree
4. Mirror the tree
5. Search for a value
6. Display tree
7. Exit
Enter your choice: 1
Enter value to insert into the tree: 9
Node 9 inserted successfully!

Menu:
1. Insert new node
2. Find number of nodes in the longest path (depth)
3. Find minimum data value in the tree
4. Mirror the tree
5. Search for a value
6. Display tree
7. Exit
Enter your choice: 2
Number of nodes in the longest path (depth): 4

Menu:
1. Insert new node
2. Find number of nodes in the longest path (depth)
3. Find minimum data value in the tree
4. Mirror the tree
5. Search for a value
6. Display tree
7. Exit
Enter your choice: 3
Minimum value in the tree: 1

Menu:
1. Insert new node
2. Find number of nodes in the longest path (depth)
3. Find minimum data value in the tree
```

```
Activities Terminal Feb 12 15:53 student@TAEComp-01: ~/Desktop/rameshwari DSA

Menu:
1. Insert new node
2. Find number of nodes in the longest path (depth)
3. Find minimum data value in the tree
4. Mirror the tree
5. Search for a value
6. Display tree
7. Exit
Enter your choice: 3
Minimum value in the tree: 1

Menu:
1. Insert new node
2. Find number of nodes in the longest path (depth)
3. Find minimum data value in the tree
4. Mirror the tree
5. Search for a value
6. Display tree
7. Exit
Enter your choice: 4
Tree mirrored successfully!

Menu:
1. Insert new node
2. Find number of nodes in the longest path (depth)
3. Find minimum data value in the tree
4. Mirror the tree
5. Search for a value
6. Display tree
7. Exit
Enter your choice: 6
Inorder traversal of the tree: 9 7 4 1

Menu:
1. Insert new node
2. Find number of nodes in the longest path (depth)
3. Find minimum data value in the tree
4. Mirror the tree
5. Search for a value
6. Display tree
7. Exit
Enter your choice: 7
Exiting program!
student@TAEComp-01:~/Desktop/rameshwari DSA$
```