**Assignment-3**

**Name of Student: Ayush Sanjay Dhangar**

**Batch: 02 Class: SY\_IT-A Roll No: 42**

**PRN: 12210406**

**Subject: IT2265 Advanced Data Structures**

**Problem Statement:** Write C or C++ program to demonstrate various tree operations (count number of nodes, leaf nodes, printing leaf nodes, height of tree, mirror image of the tree)

**Code:**

#include <stdlib.h>

struct Node {

    int data;

    struct Node\* left;

    struct Node\* right;

};

struct Node\* createNode(int value) {

    struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

    newNode->data = value;

    newNode->left = newNode->right = NULL;

    return newNode;

}

struct Node\* insertNode(struct Node\* root, int value) {

    if (root == NULL) {

        return createNode(value);

    }

    if (value < root->data) {

        root->left = insertNode(root->left, value);

    } else if (value > root->data) {

        root->right = insertNode(root->right, value);

    }

    return root;

}

int findHeight(struct Node\* root) {

    if (root == NULL) {

        return -1;

    }

    int leftHeight = findHeight(root->left);

    int rightHeight = findHeight(root->right);

    return 1 + (leftHeight > rightHeight ? leftHeight : rightHeight);

}

int countNodes(struct Node\* root) {

    if (root == NULL) {

        return 0;

    }

    return 1 + countNodes(root->left) + countNodes(root->right);

}

int countLeafNodes(struct Node\* root) {

    if (root == NULL) {

        return 0;

    }

    if (root->left == NULL && root->right == NULL) {

        return 1;

    }

    return countLeafNodes(root->left) + countLeafNodes(root->right);

}

struct Node\* createMirror(struct Node\* root) {

    if (root == NULL) {

        return NULL;

    }

    struct Node\* temp = root->left;

    root->left = createMirror(root->right);

    root->right = createMirror(temp);

    return root;

}

void printLeafNodes(struct Node\* root) {

    if (root == NULL) {

        return;

    }

    if (root->left == NULL && root->right == NULL) {

        printf("%d ", root->data);

    }

    printLeafNodes(root->left);

    printLeafNodes(root->right);

}

void printMirror(struct Node\* root) {

    if (root != NULL) {

        printMirror(root->right);

        printf("%d ", root->data);

        printMirror(root->left);

    }

}

int main() {

    struct Node\* root = NULL;

    int choice, value;

    do {

        printf("\n1. Insert Element\n");

        printf("2. Find Height\n");

        printf("3. Count Nodes\n");

        printf("4. Count Leaf Nodes\n");

        printf("5. Create Mirror Image\n");

        printf("6. Print Leaf Nodes\n");

        printf("0. Exit\n");

        printf("\nEnter your choice: ");

        scanf("%d", &choice);

        switch (choice) {

            case 1:

                printf("Enter the value to insert: ");

                scanf("%d", &value);

                root = insertNode(root, value);

                break;

            case 2:

                printf("Height of the BST: %d\n", findHeight(root));

                break;

            case 3:

                printf("Number of nodes in the BST: %d\n", countNodes(root));

                break;

            case 4:

                printf("Number of leaf nodes in the BST: %d\n", countLeafNodes(root));

                break;

            case 5:

                root = createMirror(root);

                printf("Mirror Image created.\n");

                printMirror(root);

                break;

            case 6:

                printf("Leaf Nodes: ");

                printLeafNodes(root);

                printf("\n");

                break;

            case 0:

                printf("Exiting the program.\n");

                break;

            default:

                printf("Invalid choice. Please enter a valid option.\n");

        }

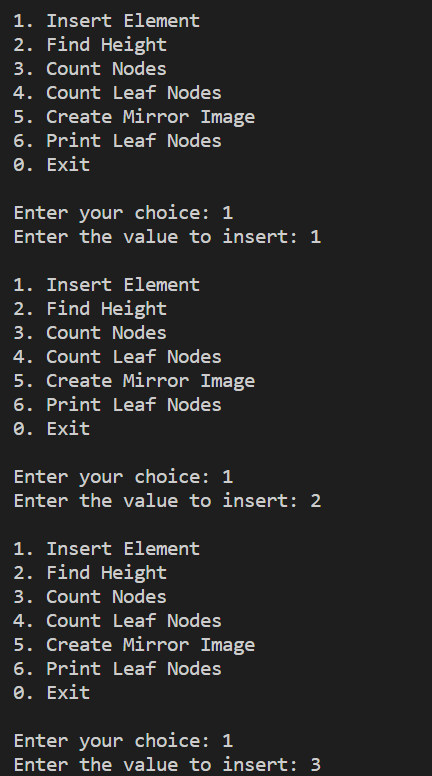
    } while (choice != 0);

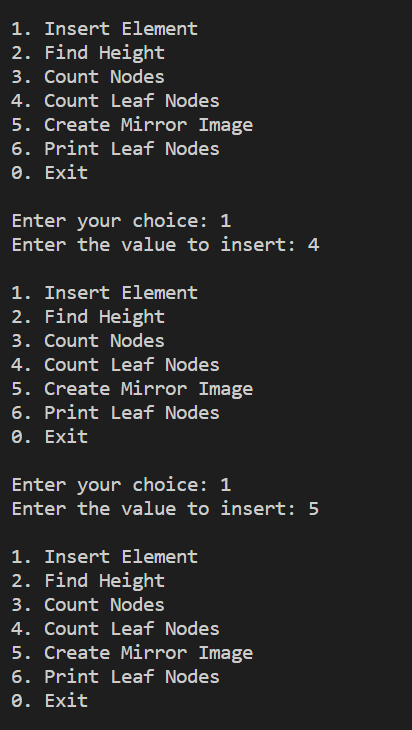
    return 0;

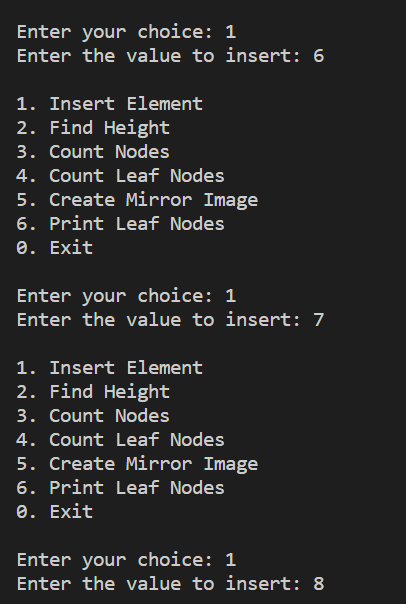
}

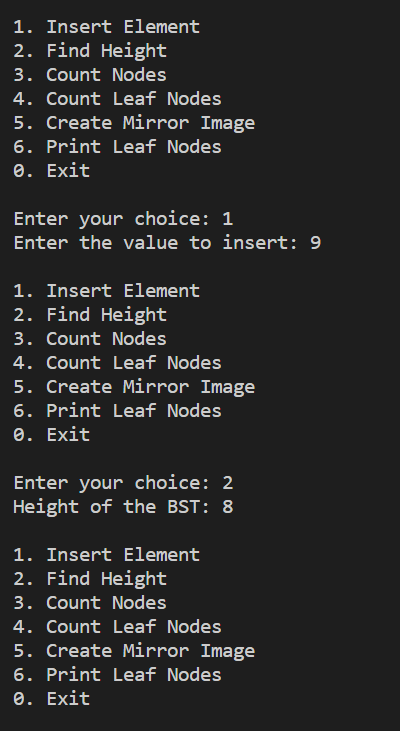
**Result:**

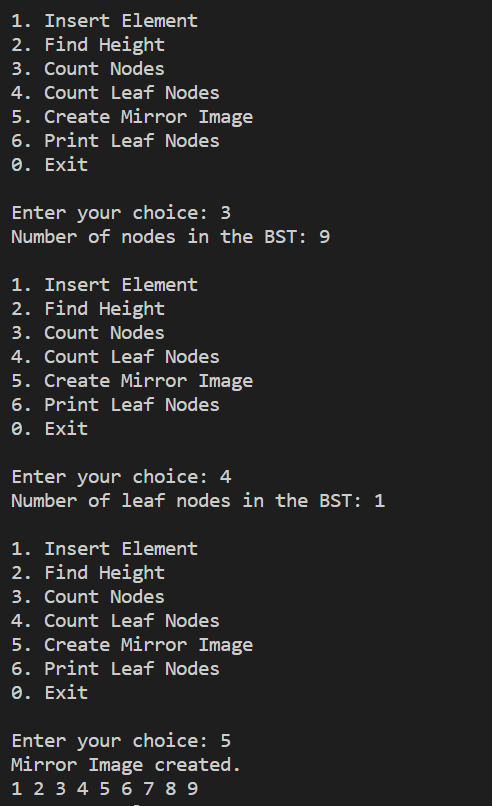
**Actual Output:**

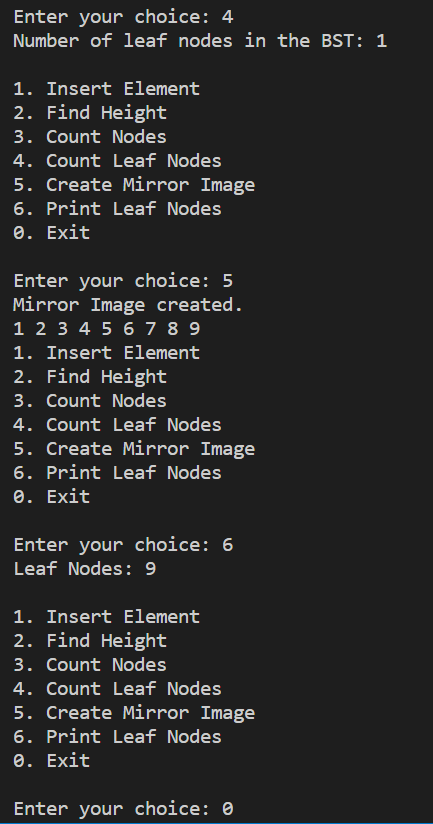
****

****

****

****

****

****