DS Lab Assignment 7

Pranav Joshi
CS-B Batch 2
Roll no: 43

Title: WAP to perform following operations on BST.

a. Create

b. Insert

c. Delete

d. Mirror Image

e. Level wise Display

f. Height of the tree

g. Display Leaf Nodes.

Code:

```c
#include <stdio.h>
#include <stdlib.h>

struct node {
    int data;
    struct node* left;
    struct node* right;
};

struct node* createNode(int data) {
    struct node* newNode = (struct node*) malloc(sizeof(struct node));
    newNode->data = data;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}

struct node* insert(struct node* root, int data) {
    if (root == NULL) {
```

```
        return createNode(data);
    }
    if (data < root->data) {
        root->left = insert(root->left, data);
    } else if (data > root->data) {
        root->right = insert(root->right, data);
    }
    return root;
}

struct node* delete(struct node* root, int data) {
    if (root == NULL) {
        return root;
    }
    if (data < root->data) {
        root->left = delete(root->left, data);
    } else if (data > root->data) {
        root->right = delete(root->right, data);
    } else {

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

        else if (root->left == NULL) {
            struct node* temp = root;
            root = root->right;
            free(temp);
        } else if (root->right == NULL) {
            struct node* temp = root;
            root = root->left;
            free(temp);
        }

        else {
            struct node* temp = root->right;
            while (temp->left != NULL) {
                temp = temp->left;
            }
            root->data = temp->data;
            root->right = delete(root->right, temp->data);
        }
    }
    return root;
}

struct node* mirror(struct node* root) {
```

```
    if (root == NULL) {
        return root;
    }
    struct node* temp = root->left;
    root->left = root->right;
    root->right = temp;
    root->left = mirror(root->left);
    root->right = mirror(root->right);
    return root;
}

void displayLevel(struct node* root, int level) {
    if (root == NULL) {
        return;
    }
    if (level == 1) {
        printf("%d ", root->data);
    } else if (level > 1) {
        displayLevel(root->left, level - 1);
        displayLevel(root->right, level - 1);
    }
}

int getHeight(struct node* root) {
    if (root == NULL) {
        return 0;
    }
    int leftHeight = getHeight(root->left);
    int rightHeight = getHeight(root->right);
    if (leftHeight > rightHeight) {
        return leftHeight + 1;
    } else {
        return rightHeight + 1;
    }
}

void displayLeaf(struct node* root) {
    if (root == NULL) {
        return;
    }
    if (root->left == NULL && root->right == NULL) {
        printf("%d ", root->data);
    }
    displayLeaf(root->left);
    displayLeaf(root->right);
}

int main() {
```

```
    struct node* root = NULL;
    root = insert(root, 80);
    insert(root, 70);
    insert(root, 90);
    insert(root, 60);
    insert(root, 75);
    insert(root, 85);
    insert(root, 95);

// the considered tree based on user input is:
            //         80
            //       /     \
            //     70      90
            //    /  \    /   \
            //   60  75  85   95

    int choice, data, height;
    while (1) {
        printf("\nBST Operations:\n");
        printf("1. Insert\n");
        printf("2. Delete\n");
        printf("3. Mirror Image\n");
        printf("4. Level wise Display\n");
        printf("5. Height of the tree\n");
        printf("6. Display Leaf Nodes\n");
        printf("7. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter data to insert: ");
                scanf("%d", &data);
                root = insert(root, data);
                printf("Node with data %d inserted in BST.\n", data);
                break;
            case 2:
                printf("Enter data to delete: ");
                scanf("%d", &data);
                root = delete(root, data);
                printf("Node with data %d deleted from BST.\n", data);
                break;
            case 3:
                root = mirror(root);
                printf("Mirror image of BST created.\n");
                height = getHeight(root);
                printf("The created Mirror image is:\n");
                for (int i = 1; i <= height; i++) {
                    displayLevel(root, i);
```

```
                }
                break;
            case 4:
                height = getHeight(root);
                printf("Level wise display of BST:\n");
                for (int i = 1; i <= height; i++) {
                    displayLevel(root, i);
                }
                printf("\n");
                break;
            case 5:
                height = getHeight(root);
                printf("Height of BST is %d.\n", height);
                break;
            case 6:
                printf("Leaf nodes of BST are: ");
                displayLeaf(root);
                printf("\n");
                break;
            case 7:
                printf("Exiting...\n");
                exit(0);
            default:
                printf("Invalid choice, please try again.\n");
                break;
        }
    }
    return 0;
}
```

Output:

Display:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS C:\Code\C\Code> cd "c:\Code\C\Code\" ; if ($?) { gcc Sem4Assignment7.c -o Sem4Assignment7 } ; if ($?) { .\Sem4Assignment7 }

BST Operations:
1. Insert
2. Delete
3. Mirror Image
4. Level wise Display
5. Height of the tree
6. Display Leaf Nodes
7. Exit
Enter your choice: 4
Level wise display of BST:
80 70 90 60 75 85 95
```

Insert:

```
BST Operations:
1. Insert
2. Delete
3. Mirror Image
4. Level wise Display
5. Height of the tree
6. Display Leaf Nodes
7. Exit
Enter your choice: 1
Enter data to insert: 50
Node with data 50 inserted in BST.

BST Operations:
1. Insert
2. Delete
3. Mirror Image
4. Level wise Display
5. Height of the tree
6. Display Leaf Nodes
7. Exit
Enter your choice: 4
Level wise display of BST:
80 70 90 60 75 85 95 50
```

Delete:

```
BST Operations:
1. Insert
2. Delete
3. Mirror Image
4. Level wise Display
5. Height of the tree
6. Display Leaf Nodes
7. Exit
Enter your choice: 2
Enter data to delete: 50
Node with data 50 deleted from BST.

BST Operations:
1. Insert
2. Delete
3. Mirror Image
4. Level wise Display
5. Height of the tree
6. Display Leaf Nodes
7. Exit
Enter your choice: 4
Level wise display of BST:
80 70 90 60 75 85 95
```

Height of the tree:

```
BST Operations:
1. Insert
2. Delete
3. Mirror Image
4. Level wise Display
5. Height of the tree
6. Display Leaf Nodes
7. Exit
Enter your choice: 5
Height of BST is 3.
```

Display Leaf Nodes:

```
BST Operations:
1. Insert
2. Delete
3. Mirror Image
4. Level wise Display
5. Height of the tree
6. Display Leaf Nodes
7. Exit
Enter your choice: 6
Leaf nodes of BST are: 60 75 85 95
```

Mirror Image:

```
BST Operations:
1. Insert
2. Delete
3. Mirror Image
4. Level wise Display
5. Height of the tree
6. Display Leaf Nodes
7. Exit
Enter your choice: 3
Mirror image of BST created.
The created Mirror image is:
80 90 70 95 85 75 60
```