DATA STRUCTURES

Division	**CS(AIML) -A**
Batch	**2**
GR-no	**12311493**
Roll no	**54**
Name	**Atharva Kangralkar**

Assignment 10:

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:-

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

struct Node {
  int data;
  struct Node *left, *right;
};

struct Node* createNode(int data) {
  struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
  newNode->data = data;
  newNode->left = 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* findMin(struct Node* node) {
  while (node->left != NULL)
```

```
    node = node->left;
  return node;
}

struct Node* deleteNode(struct Node* root, int key) {
  if (root == NULL)
    return root;
  if (key < root->data)
    root->left = deleteNode(root->left, key);
  else if (key > root->data)
    root->right = deleteNode(root->right, key);
  else {
    if (root->left == NULL) {
      struct Node* temp = root->right;
      free(root);
      return temp;
    } else if (root->right == NULL) {
      struct Node* temp = root->left;
      free(root);
      return temp;
    }
    struct Node* temp = findMin(root->right);
    root->data = temp->data;
    root->right = deleteNode(root->right, temp->data);
  }
  return root;
}
```

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

int height(struct Node* root) {
   if (root == NULL)
      return 0;
   int l = height(root->left);
   int r = height(root->right);
   return (l > r ? l : r) + 1;
}

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

void printLevel(struct Node* root, int level) {
```

```
    if (root == NULL)
        return;
    if (level == 1)
        printf("%d ", root->data);
    else {
        printLevel(root->left, level - 1);
        printLevel(root->right, level - 1);
    }
}

void levelOrder(struct Node* root) {
    int h = height(root);
    for (int i = 1; i <= h; i++) {
        printLevel(root, i);
        printf("\n");
    }
}

int main() {
    struct Node* root = NULL;
    int choice, val;

    while (1) {
        printf("\nMenu:\n1.Insert\n2.Delete\n3.Mirror\n4.Level-wise
Display\n5.Height\n6.Display Leaf Nodes\n7.Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
```

```
switch (choice) {
case 1:
   printf("Enter value to insert: ");
   scanf("%d", &val);
   root = insert(root, val);
   break;
case 2:
   printf("Enter value to delete: ");
   scanf("%d", &val);
   root = deleteNode(root, val);
   break;
case 3:
   mirror(root);
   printf("BST mirrored.\n");
   break;
case 4:
   printf("Level-wise display:\n");
   levelOrder(root);
   break;
case 5:
   printf("Height of tree: %d\n", height(root));
   break;
case 6:
   printf("Leaf nodes: ");
   displayLeafNodes(root);
   printf("\n");
   break;
case 7:
```

```
            exit(0);
        default:
            printf("Invalid choice.\n");
        }
    }

    return 0;
}
```

Output:-

```
Menu:
1.Insert
2.Delete
3.Mirror
4.Level-wise Display
5.Height
6.Display Leaf Nodes
7.Exit
Enter your choice: 1
Enter value to insert: 10

Menu:
1.Insert
2.Delete
3.Mirror
4.Level-wise Display
5.Height
6.Display Leaf Nodes
7.Exit
Enter your choice: 1
Enter value to insert: 5
```

```
Menu:
1.Insert
2.Delete
3.Mirror
4.Level-wise Display
5.Height
6.Display Leaf Nodes
7.Exit
Enter your choice: 1
Enter value to insert: 15

Menu:
1.Insert
2.Delete
3.Mirror
4.Level-wise Display
5.Height
6.Display Leaf Nodes
7.Exit
Enter your choice: 4
Level-wise display:
10
5 15
```

```
Menu:
1.Insert
2.Delete
3.Mirror
4.Level-wise Display
5.Height
6.Display Leaf Nodes
7.Exit
Enter your choice: 5
Height of tree: 2

Menu:
1.Insert
2.Delete
3.Mirror
4.Level-wise Display
5.Height
6.Display Leaf Nodes
7.Exit
Enter your choice: 6
Leaf nodes: 5 15
```

```
1.Insert
2.Delete
3.Mirror
4.Level-wise Display
5.Height
6.Display Leaf Nodes
7.Exit
Enter your choice: 3
BST mirrored.

Menu:
1.Insert
2.Delete
3.Mirror
4.Level-wise Display
5.Height
6.Display Leaf Nodes
7.Exit
Enter your choice: 4
Level-wise display:
10
15 5

Menu:
1.Insert
2.Delete
3.Mirror
4.Level-wise Display
5.Height
```