DSA Assignment: 10

Exp 10: Implementation of AVL

Shashwat Tripathi

D10A Roll No: 60

AIM: In this experiment, we will implement AVL.

CODE:

```c
#include <stdio.h>
#include <stdlib.h>
struct node{
    struct node *left;
    int data;
    struct node *right;
    int height;
};
struct node *root = NULL;
struct node *findMax(struct node *root){
    while (root->right != NULL){
        root = root->right;
    }
    return root;
}
struct node *findMin(struct node *root){
    while (root->left != NULL){
        root = root->left;
    }
    return root;
}
int height(struct node *root){
    if (root == NULL){
        return 0;
    }
    return root->height;
}
int Max(int n1, int n2){
    return ((n1 > n2) ? n1 : n2);
}
struct node *getNewNode(int data){
    struct node *newNode;
    newNode = (struct node *)malloc(sizeof(struct node));
    newNode->data = data;
    newNode->left = NULL;
    newNode->right = NULL;
    newNode->height = 1;
    return newNode;
}
int getBalance(struct node *root){
    if (root == NULL){
        return 0;
    }
    return (height(root->left) - height(root->right));
}
struct node *rightRotate(struct node *root){
```

```
    struct node *rootLeft = root->left;
    struct node *rootLeftRight = rootLeft->right;
    rootLeft->right = root;
    root->left = rootLeftRight;
    root->height = Max(height(root->left), height(root->right)) + 1;
    rootLeft->height = Max(height(rootLeft->left),
                           height(rootLeft->right)) +
                       1;
    return rootLeft;
}
struct node *leftRotate(struct node *root){
    struct node *rootRight = root->right;
    struct node *rootRightLeft = rootRight->left;
    rootRight->left = root;
    root->right = rootRightLeft;
    root->height = Max(height(root->left), height(root->right)) + 1;
    rootRight->height = Max(height(rootRight->left), height(rootRight->right))
+ 1;
    return rootRight;
}
struct node *insert(struct node *root, int data){
    if (root == NULL){
        root = getNewNode(data);
        return root;
    }
    if (data < root->data){
        root->left = insert(root->left, data);
    }
    else if (data > root->data){
        root->right = insert(root->right, data);
    }
    else{
        return root;
    }
    root->height = Max(height(root->left), height(root->right)) + 1;
    int balance = getBalance(root);
    if ((balance > 1) && (data < root->left->data)){
        return rightRotate(root);
    }
    else if ((balance < -1) && (data > root->right->data)){
        return leftRotate(root);
    }
    else if ((balance > 1) && (data > root->left->data)){
        root->left = leftRotate(root->left);
        return rightRotate(root);
    }
    else if ((balance < -1) && (data < root->right->data)){
        root->right = rightRotate(root->right);
        return leftRotate(root);
    }
    return root;
}
struct node *delete (struct node *root, int val){
    if (root == NULL){
        return root;
    }
    else if (val < root->data){
```

```
            root->left = delete (root->left, val);
        }
        else if (val > root->data){
            root->right = delete (root->right, val);
        }
        else{
            if (root->right == NULL && root->left == NULL){
                free(root);
                root = NULL;
            }
            else if (root->right == NULL){
                struct node *temp = root;
                root = root->left;
                free(temp);
            }
            else if (root->left == NULL){
                struct node *temp = root;
                root = root->right;
                free(temp);
            }
            else{
                struct node *temp = findMin(root->right);
                root->data = temp->data;
                root->right = delete (root->right, temp->data);
            }
        }
        if (root == NULL){
            return root;
        }
        root->height = Max(height(root->left),
                           height(root->right)) +
                       1;
        int balance = getBalance(root);
        if ((balance > 1) && (getBalance(root->left) >= 0)){
            // LEFT-LEFT
            return rightRotate(root);
        }
        else if ((balance < -1) && (getBalance(root->right) <= 0)){
            // RIGHT-RIGHT
            return leftRotate(root);
        }
        else if ((balance > 1) && (getBalance(root->left) < 0)){
            // LEFT-RIGHT
            root->left = leftRotate(root->left);
            return rightRotate(root);
        }
        else if ((balance < -1) && (getBalance(root->right) > 0)){
            // RIGHT-LEFT
            root->right = rightRotate(root->right);
            return leftRotate(root);
        }
        return root;
    }
    void search(struct node *root, int val){
        if (root->data == val){
            printf("\n%d is present in the tree", val);
            return;
```

```
    }
    if ((root->right == NULL && root->left == NULL) || root == NULL){
        printf("\nNot present");
        return;
    }
    if (val <= root->data){
        search(root->left, val);
    }
    else{
        search(root->right, val);
    }
}
int countAllNodes(struct node *root){
    if (root == NULL){
        return 0;
    }
    else{
        return countAllNodes(root->left) +
               countAllNodes(root->right) + 1;
    }
}
void inOrderTraversal(struct node *root){
    if (root == NULL){
        return;
    }
    inOrderTraversal(root->left);
    printf("%d ", root->data);
    inOrderTraversal(root->right);
}

int main(){
    int data, choice, val, temp;
    printf("D10A_60_Shashwat Tripathi\n");
    printf("########################################");
    printf("\n(1) Insert");
    printf("\n(2) Delete");
    printf("\n(3) Search");
    printf("\n(4) Height");
    printf("\n(5) Inorder");
    printf("\n(6) Total number of nodes");
    printf("\n(7) Exit\n");
    printf("########################################");
    while (1){
        printf("\nEnter your choice : ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            do
            {
                printf("\nEnter data to insert(enter -1 to ): ");
                scanf("%d", &temp);
                if(temp==-1){
                    break;
                }
                else{
                root = insert(root, temp);
```

```
                printf("\n%d is inserted!", temp);}
            } while (temp != -1);
            break;
        case 2:
            printf("\nEnter a value to delete: ");
            scanf("%d", &val);
            root = delete (root, val);
            printf("\n%d is deleted!", val);
            break;
        case 3:
            printf("\nEnter a number to Search");
            scanf("%d", &data);
            search(root, data);
            break;
        case 4:
            printf("\nHeight of tree is : %d",
                   height(root));
            break;
        case 5:
            printf("\nIN-ORDER : ");
            inOrderTraversal(root);
            break;
        case 6:
            printf("\nTotal number of nodes : %d",
                   countAllNodes(root));
            break;
        case 7:
            printf("\nExit\n");
            exit(1);
            break;
        default:
            printf("\nInvalid Choice");
        }
    }
    return 0;
}
```

OUTPUT:

```
C:\Windows\System32\cmd.exe

C:\Users\shweta\Documents\Shashwat\Notepad++\DSA>DSAexp10
D10A_60_Shashwat Tripathi
#######################################
(1) Insert
(2) Delete
(3) Search
(4) Height
(5) Inorder
(6) Total number of nodes
(7) Exit
#######################################
Enter your choice : 1

Enter data to insert(enter -1 to ): 12

12 is inserted!
Enter data to insert(enter -1 to ): 23

23 is inserted!
Enter data to insert(enter -1 to ): 34

34 is inserted!
Enter data to insert(enter -1 to ): 45

45 is inserted!
Enter data to insert(enter -1 to ): 56

56 is inserted!
Enter data to insert(enter -1 to ): 67

67 is inserted!
Enter data to insert(enter -1 to ): -1

Enter your choice : 4

Height of tree is : 3
Enter your choice : 5
```

```
C:\Windows\System32\cmd.exe
IN-ORDER : 12 23 34 45 56 67
Enter your choice : 6

Total number of nodes : 6
Enter your choice : 2

Enter a value to delete: 56

56 is deleted!
Enter your choice : 3

Enter a number to Search 56

Not present
Enter your choice : 7

Exit

C:\Users\shweta\Documents\Shashwat\Notepad++\DSA>
```