Lab 7 : Dipesh Singh – 190905520

Question 1 : Modify the solved exercise to find the balance factor for every node in the binary search tree.

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

typedef struct node
{
    int val, bf;
    struct node *left;
    struct node *right;
} * Node;

void inorder(Node first)
{
    if (first)
    {
        inorder(first->left);
        printf("{%d BF : %d}", first->val, first->bf);
        inorder(first->right);
    }
}
void preorder(Node first)
{
    if (first)
    {
        printf("{%d BF : %d}", first->val, first->bf);
        preorder(first->left);
        preorder(first->right);
    }
}
void postorder(Node first)
{
    if (first)
    {
        postorder(first->left);
        postorder(first->right);
        printf("{%d BF : %d}", first->val, first->bf);
    }
}

int max(int a, int b)
{
    return a > b ? a : b;
}

int height(Node n)
```

```
{
    if (n == NULL)
    {
        return -1;
    }
    return max(height(n->left), height(n->right)) + 1;
}

int BF(Node n)
{
    return height(n->left) - height(n->right);
}

void updateBF(Node head)
{
    if (head)
    {
        head->bf = BF(head);
        updateBF(head->left);
        updateBF(head->right);
    }
}

void insert(Node *head, int val)
{
    Node n = (Node)malloc(sizeof(struct node));
    n->val = val;
    n->left = n->right = NULL;
    if (*head == NULL)
    {
        *head = n;
        updateBF(*head);
        printf("**inserted**\n");
        return;
    }
    Node cur = *head;
    Node prev = NULL;
    while (cur)
    {
        prev = cur;
        if (cur->val > val)
        {
            cur = cur->left;
        }
        else if (cur->val < val)
        {
            cur = cur->right;
        }
```

```
        else
        {
            printf("**cannot insert duplicate element**\n");
            return;
        }
    }
    if (prev->val > val)
    {
        prev->left = n;
        updateBF(*head);
        printf("**inserted**\n");
        return;
    }
    else
    {
        prev->right = n;
        updateBF(*head);
        printf("**inserted**\n");
        return;
    }
}

int main()
{
    Node first = NULL;
    int choice = 0, ele;
    while (choice < 5)
    {
        printf("Please enter your choice : 1) Insert Element 2) Print Ino
rder 3) Print Preorder 4) Print Postorder 5) Exit\n :  ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            printf("Enter the element to be inserted : ");
            scanf("%d", &ele);
            insert(&first, ele);
            break;
        case 2:
            printf("The inorder is : ");
            inorder(first);
            printf("\n");
            break;
        case 3:
            printf("The preorder is : ");
            preorder(first);
            printf("\n");
            break;
```

```
        case 4:
            printf("The postorder is : ");
            postorder(first);
            printf("\n");
            break;
        case 5:
            continue;
            break;
        default:
            printf("Invalid key, try again\n");
            choice = 0;
        }
    }
    return 0;
}
```

```
dops@LAPTOP-LDOMDPE4:/mnt/d/Google Drive/Work/Study Material/2nd Year/4th Semester/DAA/DAA/Lab/Lab 7$ make balance
make: 'balance' is up to date.
dops@LAPTOP-LDOMDPE4:/mnt/d/Google Drive/Work/Study Material/2nd Year/4th Semester/DAA/DAA/Lab/Lab 7$ ./balance
Please enter your choice : 1) Insert Element 2) Print Inorder 3) Print Preorder 4) Print Postorder 5) Exit
 :  1
Enter the element to be inserted : 1
**inserted**
Please enter your choice : 1) Insert Element 2) Print Inorder 3) Print Preorder 4) Print Postorder 5) Exit
 :  1
Enter the element to be inserted : 2
**inserted**
Please enter your choice : 1) Insert Element 2) Print Inorder 3) Print Preorder 4) Print Postorder 5) Exit
 :  1
Enter the element to be inserted : 3
**inserted**
Please enter your choice : 1) Insert Element 2) Print Inorder 3) Print Preorder 4) Print Postorder 5) Exit
 :  1
Enter the element to be inserted : 4
**inserted**
Please enter your choice : 1) Insert Element 2) Print Inorder 3) Print Preorder 4) Print Postorder 5) Exit
 :  2
The inorder is : {1 BF : -3}{2 BF : -2}{3 BF : -1}{4 BF : 0}
Please enter your choice : 1) Insert Element 2) Print Inorder 3) Print Preorder 4) Print Postorder 5) Exit
 :  5
dops@LAPTOP-LDOMDPE4:/mnt/d/Google Drive/Work/Study Material/2nd Year/4th Semester/DAA/DAA/Lab/Lab 7$
```

```
dops@LAPTOP-LDOMDPE4:/mnt/d/Google Drive/Work/Study Material/2nd Year/4th Semester/DAA/DAA/Lab/Lab 7$ ./balance
Please enter your choice : 1) Insert Element 2) Print Inorder 3) Print Preorder 4) Print Postorder 5) Exit
 :  1
Enter the element to be inserted : 2
**inserted**
Please enter your choice : 1) Insert Element 2) Print Inorder 3) Print Preorder 4) Print Postorder 5) Exit
 :  1
Enter the element to be inserted : 1
**inserted**
Please enter your choice : 1) Insert Element 2) Print Inorder 3) Print Preorder 4) Print Postorder 5) Exit
 :  1
Enter the element to be inserted : 3
**inserted**
Please enter your choice : 1) Insert Element 2) Print Inorder 3) Print Preorder 4) Print Postorder 5) Exit
 :  2
The inorder is : {1 BF : 0}{2 BF : 0}{3 BF : 0}
Please enter your choice : 1) Insert Element 2) Print Inorder 3) Print Preorder 4) Print Postorder 5) Exit
 :  5
dops@LAPTOP-LDOMDPE4:/mnt/d/Google Drive/Work/Study Material/2nd Year/4th Semester/DAA/DAA/Lab/Lab 7$
```

Question 2 : Write a program to create the AVL tree by iterative insertion.

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

typedef struct node
{
    int val, height;
    struct node *left;
    struct node *right;
} * Node;

int max(int a, int b)
{
    return a > b ? a : b;
}

int height(Node n)
{
    if (n == NULL)
    {
        return -1;
    }
    return n->height;
}

int BF(Node n)
{
    return height(n->left) - height(n->right);
}

Node leftRotate(Node n)
{
    Node x = n->right;
    Node temp1 = x->left;

    x->left = n;
    n->right = temp1;
    n->height = max(height(n->left), height(n->right)) + 1;
    x->height = max(height(x->left), height(x->right)) + 1;
    return x;
}
Node rightRotate(Node n)
{
    Node x = n->left;
    Node temp1 = x->right;

    x->right = n;
```

```
    n->left = temp1;
    n->height = max(height(n->left), height(n->right)) + 1;
    x->height = max(height(x->left), height(x->right)) + 1;
    return x;
}

Node insert(Node head, int val)
{
    if (head == NULL)
    {
        Node n = (Node)malloc(sizeof(struct node));
        n->val = val;
        n->left = n->right = NULL;
        n->height = 0;
        return n;
    }
    if (val < head->val)
    {
        head->left = insert(head->left, val);
    }
    else if (val > head->val)
    {
        head->right = insert(head->right, val);
    }
    else
        return head;
    head->height = max(height(head->left), height(head->right)) + 1;

    int bf = BF(head);
    if (bf > 1 && val < head->left->val)
    {
        return rightRotate(head);
    }
    if (bf < -1 && val > head->right->val)
    {
        return leftRotate(head);
    }
    if (bf > 1 && val > head->left->val)
    {
        head->left = leftRotate(head->left);
        return rightRotate(head);
    }
    if (bf < -1 && val < head->right->val)
    {
        head->right = rightRotate(head->right);
        return leftRotate(head);
    }
    return head;
```

```
}

void inorder(Node first)
{
    if (first)
    {
        inorder(first->left);
        printf("{%d BF : %d}", first->val, BF(first));
        inorder(first->right);
    }
}
void preorder(Node first)
{
    if (first)
    {
        printf("{%d BF : %d}", first->val, BF(first));
        preorder(first->left);
        preorder(first->right);
    }
}
void postorder(Node first)
{
    if (first)
    {
        postorder(first->left);
        postorder(first->right);
        printf("{%d BF : %d}", first->val, BF(first));
    }
}

int main()
{
    Node first = NULL;
    int choice = 0, ele;
    while (choice < 5)
    {
        printf("Please enter your choice : 1) Insert Element 2) Print Ino
rder 3) Print Preorder 4) Print Postorder 5) Exit\n :  ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            printf("Enter the element to be inserted : ");
            scanf("%d", &ele);
            first = insert(first, ele);
            break;
        case 2:
            printf("The inorder is : ");
```

```
            inorder(first);
            printf("\n");
            break;
        case 3:
            printf("The preorder is : ");
            preorder(first);
            printf("\n");
            break;
        case 4:
            printf("The postorder is : ");
            postorder(first);
            printf("\n");
            break;
        case 5:
            continue;
            break;
        default:
            printf("Invalid key, try again\n");
            choice = 0;
        }
    }
    return 0;
}
```

```
dops@LAPTOP-LDOMDPE4:/mnt/d/Google Drive/Work/Study Material/2nd Year/4th Semester/DAA/DAA/Lab/Lab 7$ make avl
make: 'avl' is up to date.
dops@LAPTOP-LDOMDPE4:/mnt/d/Google Drive/Work/Study Material/2nd Year/4th Semester/DAA/DAA/Lab/Lab 7$ ./avl
Please enter your choice : 1) Insert Element 2) Print Inorder 3) Print Preorder 4) Print Postorder 5) Exit
 :  1
Enter the element to be inserted : 1
Please enter your choice : 1) Insert Element 2) Print Inorder 3) Print Preorder 4) Print Postorder 5) Exit
 :  1
Enter the element to be inserted : 2
Please enter your choice : 1) Insert Element 2) Print Inorder 3) Print Preorder 4) Print Postorder 5) Exit
 :  1
Enter the element to be inserted : 3
Please enter your choice : 1) Insert Element 2) Print Inorder 3) Print Preorder 4) Print Postorder 5) Exit
 :  2
The inorder is : {1 BF : 0}{2 BF : 0}{3 BF : 0}
Please enter your choice : 1) Insert Element 2) Print Inorder 3) Print Preorder 4) Print Postorder 5) Exit
 :  1
Enter the element to be inserted : 3
Please enter your choice : 1) Insert Element 2) Print Inorder 3) Print Preorder 4) Print Postorder 5) Exit
 :  1
Enter the element to be inserted : 4
Please enter your choice : 1) Insert Element 2) Print Inorder 3) Print Preorder 4) Print Postorder 5) Exit
 :  1
Enter the element to be inserted : 5
Please enter your choice : 1) Insert Element 2) Print Inorder 3) Print Preorder 4) Print Postorder 5) Exit
 :  2
The inorder is : {1 BF : 0}{2 BF : -1}{3 BF : 0}{4 BF : 0}{5 BF : 0}
Please enter your choice : 1) Insert Element 2) Print Inorder 3) Print Preorder 4) Print Postorder 5) Exit
 :  1
Enter the element to be inserted : 6
Please enter your choice : 1) Insert Element 2) Print Inorder 3) Print Preorder 4) Print Postorder 5) Exit
 :  2
The inorder is : {1 BF : 0}{2 BF : 0}{3 BF : 0}{4 BF : 0}{5 BF : -1}{6 BF : 0}
Please enter your choice : 1) Insert Element 2) Print Inorder 3) Print Preorder 4) Print Postorder 5) Exit
 :  5
dops@LAPTOP-LDOMDPE4:/mnt/d/Google Drive/Work/Study Material/2nd Year/4th Semester/DAA/DAA/Lab/Lab 7$
```

The operations on AVL trees take O(h) time where h is the height of the tree. Now, h = logn, hence the operations take O(logn) time.