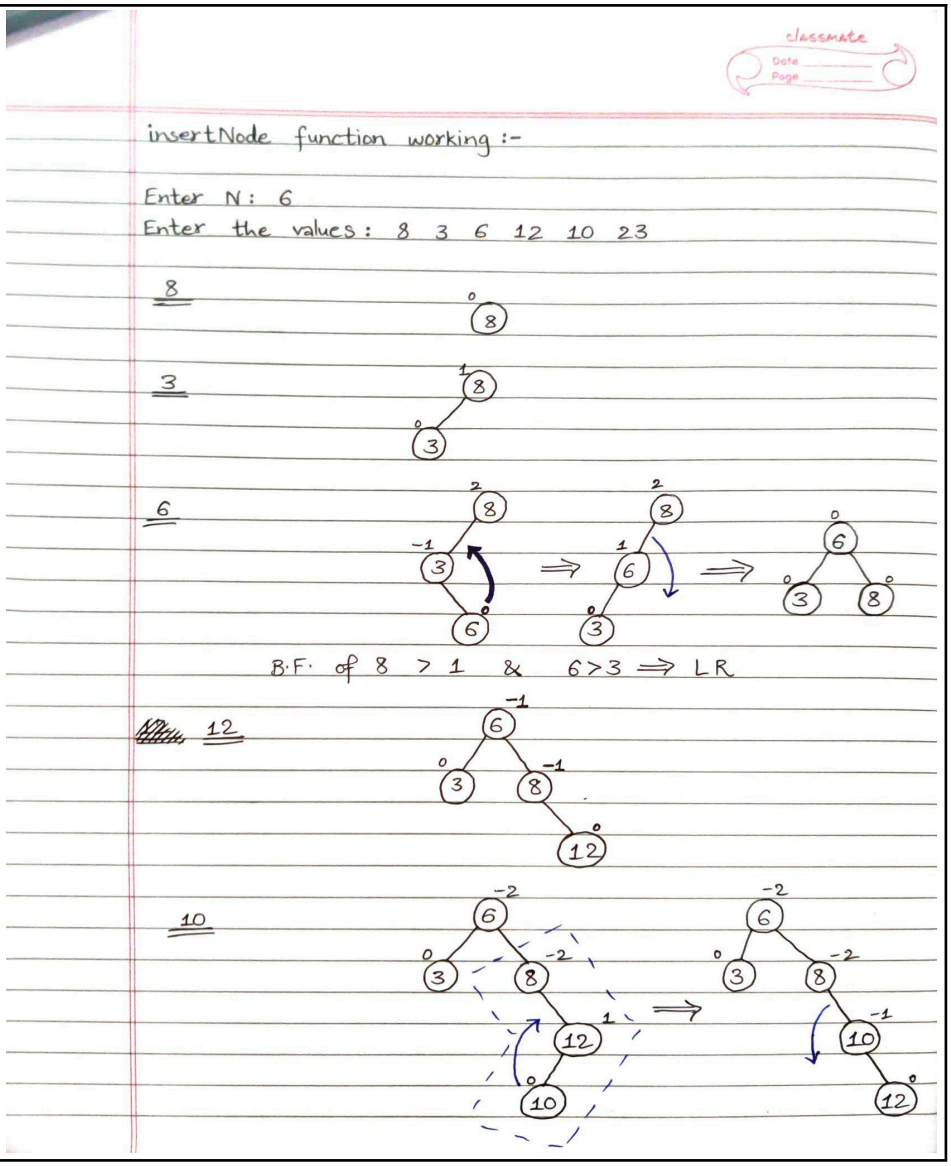
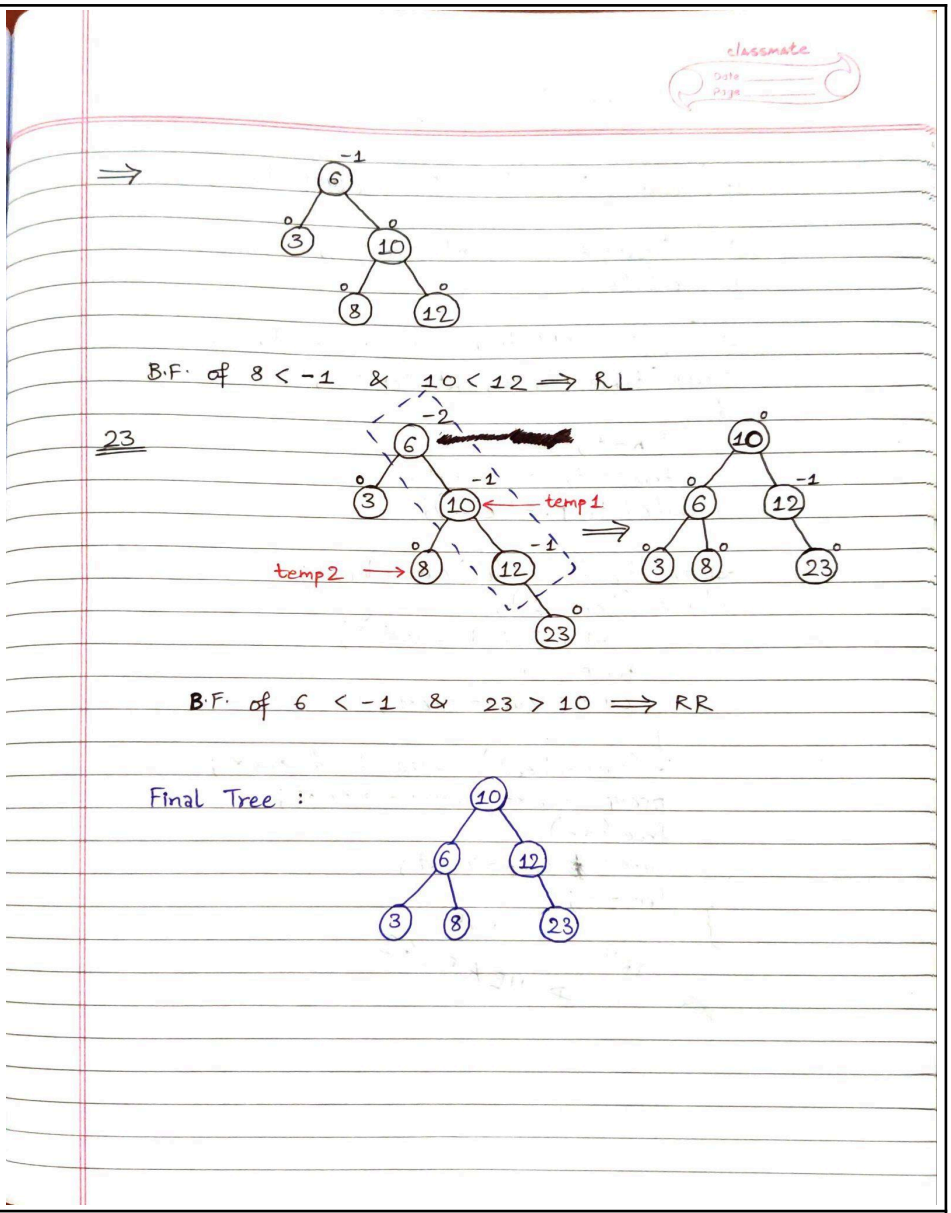
Bharatiya Vidya Bhavan's
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Autonomous Institute Affiliated to University of Mumbai)
Munshi Nagar, Andheri (W), Mumbai – 400 058.
Department of AI-ML

Experiment	7
Aim	Implement the given problem statement
Objective	Insertion in AVL Tree : Given an AVL tree and N values to be inserted in the tree. Write a function to insert elements into the given AVL tree. Note: The tree will be checked after each insertion. If it violates the properties of balanced BST, an error message will be printed followed by the inorder traversal of the tree at that moment. If instead all insertions are successful, inorder traversal of the tree will be printed. Sample input - Input : N = 3 Values to be inserted = {5,1,4} Output : 1 4 5
Name	Balla Mahadev Shrikrishna
UCID	2023300010
Class	A
Batch	A
Date of Submission	10-10-24

Explanation of the technique used	



Program(Code)

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

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

Node *getNode(int val){
    Node *new = (Node *)malloc(sizeof(Node));
    new->left = new->right = NULL;
    new->val = val;
    new->height = 1;
    return new;
}
```

```
int max(int a, int b){
   if(a>b) return a;
   return b;
}

int getHeight(Node *node){
   if(node==NULL) return 0;
   return node->height;
}

void inOrderTraversal(Node *root){
   if(root!=NULL){
      inOrderTraversal(root->left);
      printf("%d  ", root->val);
      inOrderTraversal(root->right);
   }
}

Node *RR(Node *node){
   Node *temp1 = node->right;
   Node *temp2 = temp1->left;

   temp1->left = node;
   node->right = temp2;

   node->height = max(getHeight(node->left), getHeight(node->right))
+ 1;
   temp1->height = max(getHeight(temp1->left),
getHeight(temp1->right)) + 1;

   return temp1;
}

Node *LL(Node *node){
   Node *temp1 = node->left;
   Node *temp2 = temp1->right;

   temp1->right = node;
   node->left = temp2;

   node->height = max(getHeight(node->left), getHeight(node->right))
+ 1;
   temp1->height = max(getHeight(temp1->left),
getHeight(temp1->right)) + 1;

   return temp1;
}

Node *LR(Node *node){
```

```
    node->left = RR(node->left);
    return LL(node);
}

Node *RL(Node *node){
    node->right = LL(node->right);
    return RR(node);
}

Node* insertNode(Node *root, int val){
    if(root == NULL) return getNode(val);

    if(val > root->val){
        root->right = insertNode(root->right, val);
    }
    else if(val < root->val){
        root->left = insertNode(root->left, val);
    }
    else{
        return root;
    }

    root->height = max(getHeight(root->left), getHeight(root->right)) +
1;
    int bal = getHeight(root->left) - getHeight(root->right);

    if(bal > 1  &&  val < root->left->val){
        printf("Left-Left Case...Inorder Traversal at this moment : ");
        inOrderTraversal(root);
        printf("\n");
        return LL(root);
    }
    if(bal < -1  &&  val > root->right->val){
        printf("Right-Right Case...Inorder Traversal at this moment : ");
        inOrderTraversal(root);
        printf("\n");
        return RR(root);
    }
    if(bal > 1  &&  val > root->left->val){
        printf("Left-Right Case...Inorder Traversal at this moment : ");
        inOrderTraversal(root);
        printf("\n");
        return LR(root);
    }
    if(bal < -1  &&  val < root->right->val){
        printf("Right-Left Case...Inorder Traversal at this moment : ");
        inOrderTraversal(root);
        printf("\n");
        return RL(root);
    }
```

```
    return root;
}

void freeTree(Node *root){
    if(root!=NULL){
        freeTree(root->left);
        freeTree(root->right);
        free(root);
    }
}

int main(){
    Node *root = NULL;
    int t, val;
    printf("Enter N : ");
    scanf("%d", &t);
    printf("Enter the values : ");
    while(t--){
        scanf("%d", &val);
        root = insertNode(root, val);
    }
    printf("Inorder Traversal : ");
    inOrderTraversal(root);
    printf("\n");
    freeTree(root);
    return 0;
}
```

Output

avl.c

```c
void freeTree(Node *root){
        freeTree(root->left);
        freeTree(root->right);
        free(root);
    }
}

int main(){
    Node *root = NULL;
    int t, val;
    printf("Enter N : ");
    scanf("%d", &t);
    printf("Enter the values : ");
    while(t--){
        scanf("%d", &val);
        root = insertNode(root, val);
    }
    printf("Inorder Traversal : ");
    inOrderTraversal(root);
    printf("\n");
    freeTree(root);
    return 0;
}
```

```
PS C:\Mahadev\S.E\DS\Lab Sessions> gcc avl.c
PS C:\Mahadev\S.E\DS\Lab Sessions> ./a.exe
Enter N : 10
Enter the values : 1 5 13 28 14 7 4 9 3 2
Right-Right Case...Inorder Traversal at this moment : 1  5  13
Right-Left Case...Inorder Traversal at this moment : 13  14  28
Right-Left Case...Inorder Traversal at this moment : 1  5  7  13  14  28
Right-Left Case...Inorder Traversal at this moment : 1  3  4
Left-Left Case...Inorder Traversal at this moment : 1  2  3  4  5  7  9  13  14  28
Inorder Traversal : 1  2  3  4  5  7  9  13  14  28
PS C:\Mahadev\S.E\DS\Lab Sessions>
```

avl.c

```c
Node* insertNode(Node *root, int val){
    if(root == NULL) return getNode(val);

    if(val > root->val){
        root->right = insertNode(root->right, val);
    }
    else if(val < root->val){
        root->left = insertNode(root->left, val);
    }
    else{
        return root;
    }

    root->height = max(getHeight(root->left), getHeight(root->right)) + 1;
    int bal = getHeight(root->left) - getHeight(root->right);

    if(bal > 1  &&  val < root->left->val){
        printf("Left-Left Case...Inorder Traversal at this moment : ");
        inOrderTraversal(root);
        printf("\n");
        return LL(root);
    }
    if(bal < -1  &&  val > root->right->val){
        printf("Right-Right Case...Inorder Traversal at this moment : ");
        inOrderTraversal(root);
```

```
PS C:\Mahadev\S.E\DS\Lab Sessions> ./a.exe
Enter N : 6
Enter the values : 8 3 6 12 10 23
Left-Right Case...Inorder Traversal at this moment : 3  6  8
Right-Left Case...Inorder Traversal at this moment : 8  10  12
Right-Right Case...Inorder Traversal at this moment : 3  6  8  10  12  23
Inorder Traversal : 3  6  8  10  12  23
PS C:\Mahadev\S.E\DS\Lab Sessions>
```

Conclusion	In this implementation of an AVL tree, we leveraged single and double rotations (LL, RR, LR, RL) to maintain the tree's balance after insertions. AVL trees are self-balancing binary search trees, and maintaining balance is crucial for ensuring efficient performance of operations such as insertions and deletions.