ACPCE
Where knowledge is second nature

Jawahar Education Society's
A. C. Patil College of Engineering, Kharghar
Navi Mumbai 410210

Student Name: Chetan Ingale

Course Name: C.S.E. (IoT CS BC)

Year: S.E.

Roll No.: 17

PRN No.: 221111030

Course code: CSL301

Semester: 3

Experiment Evaluation Sheet

Experiment No.: 9

Experiment Name:
Write a program to implement AVL Trees.

Sr No.	Evaluation Criteria	Marks (Out of 9)	Performance Date	Correction Date and Signature of Instructor
1	**Experiment Performance**			
2	**Journal Performance**			
3	**Punctuality**			
Total				

Code :

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

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

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

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

Node *newNode(int key) {
    Node *node = (Node *)malloc(sizeof(Node));
    node->key = key;
    node->left = NULL;
    node->right = NULL;
    node->height = 1;
    return node;
}

Node *rightRotate(Node *y) {
    Node *x = y->left;
    Node *T2 = x->right;
    x->right = y;
    y->left = T2;
    y->height = max(height(y->left), height(y->right)) + 1;
    x->height = max(height(x->left), height(x->right)) + 1;
    return x;
}

Node *leftRotate(Node *x) {
    Node *y = x->right;
    Node *T2 = y->left;
    y->left = x;
    x->right = T2;
    x->height = max(height(x->left), height(x->right)) + 1;
    y->height = max(height(y->left), height(y->right)) + 1;
    return y;
}

int getBalance(Node *N) {
    if (N == NULL)
        return 0;
    return height(N->left) - height(N->right);
}
```

Code :

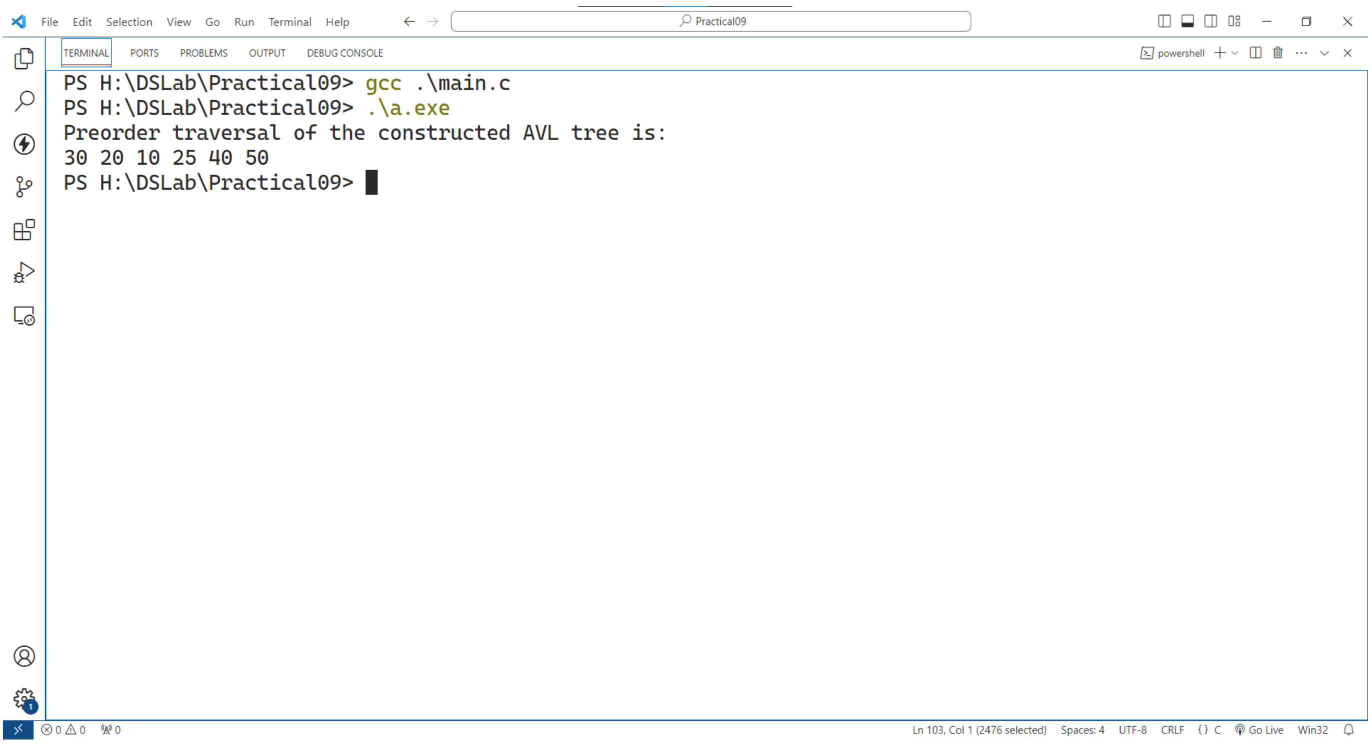
```
Node *insert(Node *node, int key) {
    if (node == NULL)
        return newNode(key);
    if (key < node->key)
        node->left = insert(node->left, key);
    else if (key > node->key)
        node->right = insert(node->right, key);
    else
        return node;
    node->height = 1 + max(height(node->left), height(node->right));
    int balance = getBalance(node);

    if (balance > 1 && key < node->left->key)
        return rightRotate(node);
    if (balance < -1 && key > node->right->key)
        return leftRotate(node);
    if (balance > 1 && key > node->left->key) {
        node->left = leftRotate(node->left);
        return rightRotate(node);
    }
    if (balance < -1 && key < node->right->key) {
        node->right = rightRotate(node->right);
        return leftRotate(node);
    }
    return node;
}

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

int main() {
    Node *root = NULL;
    root = insert(root, 10);
    root = insert(root, 20);
    root = insert(root, 30);
    root = insert(root, 40);
    root = insert(root, 50);
    root = insert(root, 25);
    printf("Preorder traversal of the constructed AVL tree is: \n");
    preOrder(root);
    return 0;
}
```

Output :



```
PS H:\DSLab\Practical09> gcc .\main.c
PS H:\DSLab\Practical09> .\a.exe
Preorder traversal of the constructed AVL tree is:
30 20 10 25 40 50
PS H:\DSLab\Practical09> █
```

Conclusion :

Through this experiment we have learnt about how to implement a AVL Tree using the C language. Various operations like insertion, balancing and traversal are applied on the AVL Tree. This experiment helps us in using AVL Tree as a data structure for further reference.