EX NO 10 : IMPLEMENTATION OF AVL TREE

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

struct AVLNode {
    int key;
    struct AVLNode *left;
    struct AVLNode *right;
    int height;
};

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

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

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

struct AVLNode *rotateRight(struct AVLNode *y) {
    struct AVLNode *x = y->left;
    struct AVLNode *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;
}
```

```
struct AVLNode *rotateLeft(struct AVLNode *x) {
    struct AVLNode *y = x->right;
    struct AVLNode *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(struct AVLNode *node) {
    if (node == NULL)
        return 0;
    return height(node->left) - height(node->right);
}

struct AVLNode *insert(struct AVLNode *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 rotateRight(node);

    if (balance < -1 && key > node->right->key)
        return rotateLeft(node);

    if (balance > 1 && key > node->left->key) {
        node->left = rotateLeft(node->left);
        return rotateRight(node);
    }
```

```
    if (balance < -1 && key < node->right->key) {
        node->right = rotateRight(node->right);
        return rotateLeft(node);
    }

    return node;
}

void inorder(struct AVLNode *node) {
    if (node != NULL) {
        inorder(node->left);
        printf("%d ", node->key);
        inorder(node->right);
    }
}

int main() {
    struct AVLNode *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("Inorder traversal of the AVL tree:\n");
    inorder(root);
    printf("\n");

    return 0;
}
```

OUTPUT:

```
------- AVL TREE --------

1. Insert
2. Delete
3. Search
4. Inorder
5. Preorder
6. Postorder
7. EXIT

Enter Your Choice: 2

Enter data: 3


Do you want to continue? yes


------- AVL TREE --------

1. Insert
2. Delete
3. Search
4. Inorder
5. Preorder
6. Postorder
7. EXIT
```