Assignment 6

Name:Soham Mahajan

Roll No:SYCOC160

Program:

```
#include <iostream>

using namespace std;


struct Node {

    int key;

    Node* left;

    Node* right;

    int height;

};


int max(int a, int b) {

    return (a > b) ? a : b;

}


int height(Node* node) {

    if (node == nullptr)

        return 0;

    return node->height;

}


Node* newNode(int key) {
    Node* node = new Node;
    node->key = key;
    node->left = nullptr;
    node->right = nullptr;
    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* node) {
    if (node == nullptr)
        return 0;
    return height(node->left) - height(node->right);
}


Node* insert(Node* node, int key) {
    if (node == nullptr)
        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);

    // Left Heavy
    if (balance > 1) {
        if (key < node->left->key)
            return rightRotate(node);
        else {
            node->left = leftRotate(node->left);
            return rightRotate(node);
        }
    }

    // Right Heavy
    if (balance < -1) {
        if (key > node->right->key)
            return leftRotate(node);
        else {
            node->right = rightRotate(node->right);
            return leftRotate(node);
        }
    }

    return node;
}
```

```
void preOrder(Node* root) {
   if (root != nullptr) {
      cout << root->key << " ";
      preOrder(root->left);
      preOrder(root->right);
   }
}


int main() {
   Node* root = nullptr;


   root = insert(root, 10);
   root = insert(root, 20);
   root = insert(root, 30);
   root = insert(root, 40);
   root = insert(root, 50);
   root = insert(root, 25);


   cout << "Preorder traversal of the AVL tree is: ";
   preOrder(root);
   cout << endl;


   return 0;
}
```

Output:

```
/tmp/1ssN1j6z3y.o
```

```
Preorder traversal of the AVL tree is: 30 20 10 25 40 50
```