Common Boilerplate Code for AVL Tree

Boilerplate Code

```cpp
#include <iostream>
using namespace std;

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

int maxVal(int a, int b) {
    if (a > b)
        return a;
    else
        return b;
}

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

Node* newNode(int key) {
    Node* node = new Node;
    node->key = key;
    node->height = 1;
    node->left = node->right = nullptr;
    return node;
}

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

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

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 + maxVal(height(node->left), height(node->right));
    int balance = getBalance(node);
    if (balance > 1) {
        if (key > node->right->key)
            return leftRotate(node);
        else {
            node->right = rightRotate(node->right);
            return leftRotate(node);
        }
    }
    if (balance < -1) {
        if (key < node->left->key)
            return rightRotate(node);
        else {
            node->left = leftRotate(node->left);
            return rightRotate(node);
        }
    }
    return node;
}
```