

Common Boilerplate Code for AVL Tree

Boilerplate Code

```
1  #include <iostream>
2  using namespace std;
3
4  struct Node {
5      int key;
6      int height;
7      Node* left;
8      Node* right;
9  };
10
11 int maxVal(int a, int b) {
12     if (a > b)
13         return a;
14     else
15         return b;
16 }
17
18 int height(Node* node) {
19     if (node == nullptr)
20         return 0;
21     else
22         return node->height;
23 }
24
25 Node* newNode(int key) {
26     Node* node = new Node;
27     node->key = key;
28     node->height = 1;
29     node->left = node->right = nullptr;
30     return node;
31 }
32
33 Node* rightRotate(Node* y) {
34     Node* x = y->left;
35     Node* T2 = x->right;
36     x->right = y;
37     y->left = T2;
38     y->height = maxVal(height(y->left), height(y->right)) + 1;
39     x->height = maxVal(height(x->left), height(x->right)) + 1;
40     return x;
41 }
42
43 Node* leftRotate(Node* x) {
44     Node* y = x->right;
45     Node* T2 = y->left;
46     y->left = x;
```

```

47     x->right = T2;
48     x->height = maxVal(height(x->left), height(x->right)) + 1;
49     y->height = maxVal(height(y->left), height(y->right)) + 1;
50     return y;
51 }
52
53 int getBalance(Node* node) {
54     if (node == nullptr)
55         return 0;
56     return height(node->right) - height(node->left);
57 }
58
59 Node* insert(Node* node, int key) {
60     if (node == nullptr)
61         return newNode(key);
62     if (key < node->key)
63         node->left = insert(node->left, key);
64     else if (key > node->key)
65         node->right = insert(node->right, key);
66     else
67         return node;
68     node->height = 1 + maxVal(height(node->left), height(node->right));
69     int balance = getBalance(node);
70     if (balance > 1) {
71         if (key > node->right->key)
72             return leftRotate(node);
73         else {
74             node->right = rightRotate(node->right);
75             return leftRotate(node);
76         }
77     }
78     if (balance < -1) {
79         if (key < node->left->key)
80             return rightRotate(node);
81         else {
82             node->left = leftRotate(node->left);
83             return rightRotate(node);
84         }
85     }
86     return node;
87 }

```