05/30/17 10:11:48 /Users/darkcloud/Desktop/CSE 330/Lab8/[Lab8] Adnar Lozano.cpp

```cpp
#include <iostream>
using namespace std;
class Node {
public:
    int key;
    int height;
    Node* left;
    Node* right;
};
int height(Node* node) {
    if (node == NULL) return 0;
    return node->height;
}
int max(int a, int b) {
    return (a > b) ? a : b;
}
Node* newNode(int key) {
    Node* node = new 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* node) {
    if (node == NULL) return 0;
    return height(node->left) - height(node->right);
}
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) {
        cout << 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);
  cout << "Printing the AVL tree\n";
  preOrder(root);
  cout << endl;
  return 0;
}
```