//Name: Mehmet Fatih Çelik

//ID: 2385268

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

struct AVLTreeNode{

int val;

int height;

struct AVLTreeNode \*left;

struct AVLTreeNode \*right;

};

typedef struct AVLTreeNode \*AVLTree;

AVLTree CreateAvlTree(AVLTree, int);

int HeightAvlTree(AVLTree);

AVLTree RightRotation(AVLTree);

AVLTree LeftRotation(AVLTree);

AVLTree DoubleRightRotation(AVLTree);

AVLTree DoubleLeftRotation(AVLTree);

int MaxOfTwo(int, int);

void PrintTree(AVLTree t);

void displayAVLTreeStructure(AVLTree, int);

int checkGuess(AVLTree, int);

int main(){

srand(time(NULL));

AVLTree root = NULL;

int i, guess, flag;

for(i=0;i<6;i++)

root = CreateAvlTree(root, rand()%101);

printf("Guess a number: ");

scanf("%d",&guess);

flag = checkGuess(root, guess);

if (flag == 1)

printf("You win!\n");

else

printf("You loose!\n");

printf("Tree structure is: ");

PrintTree(root);

//displayAVLTreeStructure(root, 0); //this function for better understanding the tree (Meryem Hoca gave the algorithm)

return 0;

}

AVLTree CreateAvlTree(AVLTree t, int num){

if (t == NULL){

t = (AVLTree)malloc(sizeof(struct AVLTreeNode));

if (t == NULL){

printf("Error occured while allocating the memory!");

exit(-1);

}

t->val = num;

t->height = 0;

t->left = NULL;

t->right = NULL;

}

else{

if (num < t->val){

t->left = CreateAvlTree(t->left, num);

if (HeightAvlTree(t->left) - HeightAvlTree(t->right) == 2){

if (num < t->left->val)

t = RightRotation(t);

else

t = DoubleRightRotation(t);

}

}

else if (num > t->val){

t->right = CreateAvlTree(t->right, num);

if (HeightAvlTree(t->right) - HeightAvlTree(t->left) == 2){

if (num > t->right->val)

t = LeftRotation(t);

else

t = DoubleLeftRotation(t);

}

}

t->height = MaxOfTwo(HeightAvlTree(t->left),HeightAvlTree(t->right))+ 1;

}

return t;

}

int HeightAvlTree(AVLTree t){

if (t == NULL)

return -1;

else

return t->height;

}

AVLTree RightRotation(AVLTree t){

AVLTree temp;

temp = t->left;

t->left = temp->right;

temp->right = t;

t->height = MaxOfTwo(HeightAvlTree(t->left),HeightAvlTree(t->right))+1;

temp->height = MaxOfTwo(HeightAvlTree(temp->left),HeightAvlTree(temp->right))+1;

return temp;

}

AVLTree LeftRotation(AVLTree t){

AVLTree temp;

temp = t->right;

t->right = temp->left;

temp->left = t;

t->height = MaxOfTwo(HeightAvlTree(t->left),HeightAvlTree(t->right))+1;

temp->height = MaxOfTwo(HeightAvlTree(temp->left),HeightAvlTree(temp->right))+1;

return temp;

}

AVLTree DoubleRightRotation(AVLTree t){

t->left = LeftRotation(t->left);

return RightRotation(t);

}

AVLTree DoubleLeftRotation(AVLTree t){

t->right = RightRotation(t->right);

return LeftRotation(t);

}

int MaxOfTwo(int a, int b){

if (a > b)

return a;

else

return b;

}

void PrintTree(AVLTree t){

if(t != NULL){

PrintTree(t->left);

printf("%d ",t->val);

PrintTree(t->right);

}

}

void displayAVLTreeStructure(AVLTree t, int depth){

int i;

if (t != NULL){

displayAVLTreeStructure(t->right, depth + 1);

for (i = 0; i < depth; i++)

printf(" ");

printf("%d\n", t->val);

displayAVLTreeStructure(t->left, depth + 1);

}

}

int checkGuess(AVLTree t, int value){

if (t == NULL)

return 0;

else if (value == t->val)

return 1;

else if (value < t->val)

return checkGuess(t->left, value);

else if (value > t->val)

return checkGuess(t->right, value);

}