typedef struct BinNode {

int key;

BinNode\* left;

BinNode\* right;

}BinNode;

BinNode\* newNode(int key) {

BinNode\* node = (BinNode\*)malloc(sizeof(BinNode));

node->key = key;

node->left = NULL;

node->right = NULL;

return node;

}

int height(BinNode\* node) {

if (node == NULL)

return -1;

int r = height(node->right), l = height(node->left);

if (r > l)

return r + 1;

return l + 1;

}

int sum(BinNode\* node) {

int sum1 = 0;

if (node == NULL)

return 0;

sum1 += node->key + sum(node->left) + sum(node->right);

return sum1;

}

int biggest(BinNode\* node) {

int l = node->key;

int r = node->key;

if (node ->left != NULL)

l = biggest(node->left);

if (node->left != NULL)

r = biggest(node->right);

if (l > r)

return l;

if (r > l)

return r;

}

int numShows(BinNode\* node, int x)

{

if (node == NULL)

return 0;

int sumShows = 0;

if (node->key == x)

sumShows++;

return sumShows += numShows(node->left, x) + numShows(node->right, x);

}

int isInTheTree(BinNode\* node, int x) {

if (node == NULL)

return 0;

if (node->key == x)

return 1;

int result = isInTheTree(node->left, x);

if (result)//אם היה כבר בחלק המשאלי ישר נחזיר 1 ולא נבדוק בכלל את החלק הימני

return 1;

result = isInTheTree(node->right, x);//זה יתבצע רק אם לא מצאנו בחלק השמאלי

return result;

}

int isInTheOtherTree(BinNode\* node1, BinNode\* node2) {

if (node2 == NULL)

return 1;

if (!isInTheTree(node1, node2->key))

return 0;

if (!isInTheOtherTree(node1, node2->left))

return 0;

if (!isInTheOtherTree(node1, node2->right))

return 0;

return 1;

}

int lessThenTree(BinNode\* node, int x) {

if (node == NULL)

return 1;

if (node->key <= x)

return 0;

if (!lessThenTree(node->left, x))

return 0;

if (!lessThenTree(node->right, x))

return 0;

return 1;

}

int treeLessThenTree(BinNode\* t1, BinNode\* t2) {// האם כל ערך בעץ 1 קטן מכל הערכים בעץ 2

if (t1 == NULL)

return 1;

if (!lessThenTree(t2, t1->key))

return 0;

if (!treeLessThenTree(t1->left, t2))

return 0;

if (!treeLessThenTree(t1->right, t2))

return 0;

return 1;

}