In each of the following questions, you will be writing functions that utilize binary trees created with the following struct:

```
struct treenode {
      int data;
      struct treenode* left;
      struct treenode* right;
};
```

**10)** Write a function that operates on a binary tree of integers. Your function should sum up the all of the odd numbers in the tree EXCEPT for the numbers in leaf nodes, which should instead be ignored. Make use of the function prototype shown below.

```
int sum_nonleaf_odd(struct treenode* p){
    int val;
    if(!root) return 0;
    val = root->data;
    if (val%2==0 || (!root->left) && !root->right))
       return sum_nonleaf_odd(root->left) + sum_nonleaf_odd)(root->right);
    else
       return val + sum_nonleaf_odd(root->left)+ sum_nonleaf_odd)(root->right);
}
```

**11)** Write a function that returns the number of leaf nodes in the binary tree pointed to by p. Utilize the function prototype provided below.

```
int numLeafNodes(struct treenode* p);
    if( p ){
       if(! p ->left && ! p ->right) return 1;
       return numLeafNodes( p ->left)+
              numLeafNodes( p ->right);
    }
    else return 0;
}
```

**12)** Write a recursive function to compute the height of a tree, defined as the length of the longest path from the root to a leaf node. For the purposes of this problem a tree with only one node has height 1 and an empty tree has height 0.

```
int height(struct treenode* root);
    if (!root) return 0;
    int lh=height(root->left);
    int rh = height(root->right);
    if (lh > rh)
       return 1+lh;
    return 1+rh;
}
```