Computer Science I – Summer 2011
Recitation #8: More Binary Search Trees (Solutions)

1. (Aug '07 CS-B-5) **a)** Indicate in few words, the purpose of the following function. The struct treenode is the same as the one defined in problem 4 on the previous page.

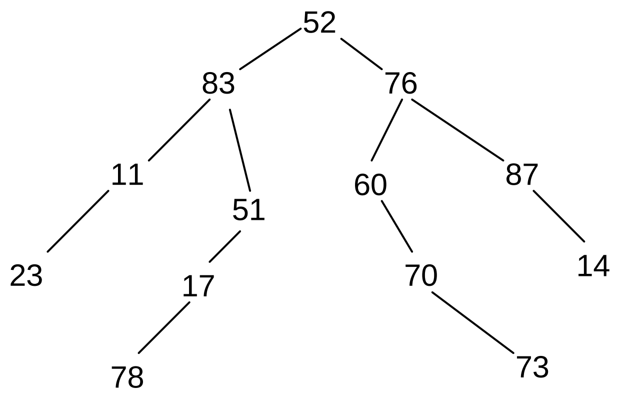
```
int f(struct treenode * p) {

    int val;
    if (p == NULL) return 0;
    val = p->data;
    if (val%2 == 0)
        return val + f(p->left) + f(p->right);
    else
        return f(p->left) + f(p->right);

}
```

Returns the sum of the nodes storing even values in the tree.

b) What does the function return, given the following tree?



Answer: **78+52+76+60+70+14 = 350**

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;
};
```

2) (Dec '08 CS-B-2) 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 sum_nonleaf_odd(struct treenode* p)
{

    if (p == NULL) return 0;

    if (p->left == NULL && p->right == NULL)
        return 0;

    int sum = 0;
    if (p->data%2 == 1)
        sum += p->data;

    sum += sum_nonleaf_odd(p->left);
    sum += sum_nonleaf_odd(p->right);
    return sum;
}
```

3) (May '08 CS-B-4) 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);

int numLeafNodes(struct treenode* p) {

    if (p == NULL) return 0;
    if (p->left == NULL &&p->right == NULL)
        return 1;

    return numLeafNodes(p->left) + numLeafNodes(p->right);

}
```

4) (Dec '07 CS-B-3) 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);

int height(struct treenode* root)
{
     int leftheight, rightheight;

     if(root == NULL)
          return 0;

     leftheight = height(root->left);
     rightheight = height(root->right);

     if(leftheight > rightheight)
          return leftheight + 1;

     return rightheight + 1;
}
```