1) (10 pts) DSN (Binary Trees)

A binary search tree is considered "lopsided" if the root's left subtree height and right subtree height differ by more than one (i.e., the left subtree is more than one level deeper or shallower than the right subtree). This is different from the definition of "balanced" that comes up in relation to AVL trees, because the "lopsided" property only applies to the root of the tree – not every single node in the tree.

Write a function, *isLopsided()*, that takes the root of a binary search tree and returns 1 if the tree is lopsided, and 0 otherwise. You may write helper functions as you see fit. The node struct and function signature are as follows:

```
typedef struct node {
   struct node *left, *right;
   int data;
} node;

int isLopsided(node *root)
{
   int lHeight, rHeight, diff;

   if (root == NULL) return 0;

   lHeight = height(root->left);
   rHeight = height(root->right);
   diff = lHeight - rHeight;

   return (diff >= 2 || diff <= -2);
}

int max(int a, int b)
{
   return (a > b) ? a : b;
}

int height(node *root)
{
   if (root == NULL) return -1;
   return 1 + max(height(root->left), height(root->right));
}
```

Grading:
2 pts for checking root == NULL in isLopsided()
2 pts for realizing they need to write a separate height() function
3 pts for the rest of the logic in isLopsided, such as the diff check (award partial credit as you deem fit)
3 pts for the logic in height() (award partial credit as you deem fit)
Note - for this particular question, if height returns 0 for null tree, that is fine since all that matters for correctness here is relative height.