1) (10 pts) DSN (Recursive Coding and Binary Trees)

Write a recursive function called *prune()* that takes the root of a binary tree and deletes all leaf nodes from the tree. Be sure to update any pointers that need to be updated to account for deleted nodes, carefully avoid any potential segmentation faults, and avoid memory leaks. (You may assume all the nodes in the tree have been dynamically allocated.)

For example:

```
               Initial Tree                           Resulting Tree
     (leaf nodes underlined and bold)                 (after pruning)

                   18                                      18
                  /  \                                       \
                25    1                                       1
                     / \                                       \
                    2   9                                       9
                         \
                          16
```

This function should return NULL if the root it receives gets deleted. Otherwise, it should return *root* itself. Note that if *root* is NULL, the function should simply return NULL without taking any further action.

You cannot write any helper functions for this problem. All your work must be contained in a single function called *prune()*. The node struct definition and function signature are as follows:

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

node *prune(node *root)
{
      if (root == NULL)      // 1 pt
            return NULL;

      if (root->left == NULL && root->right == NULL)  // 1 pt for leaf check
      {
            free(root);                                // 2 pts for freeing
            return NULL;                               // 1 pt for returning NULL
      }

      // 2 pts for setting children to appropriate values somehow
      // 2 pts for appropriate recursive calls
      root->left = prune(root->left);               // We could check for non-NULL
      root->right = prune(root->right);             // before making the recursive
                                                    // calls, but it is not strictly
      // 1 pt for returning root                    // necessary.
      return root;
}
```