A Lengthier Recursion

This is a **warmup** exercise. It is **not compulsory**, and may be completed **individually or with your lab partner**.

In the last few exercises, we've seen how recursion is a really powerful tool for adding or multiplying numbers together. (We've also seen that I have terrible choices in pets...)

Recursions work really well with linked lists, because we have two fairly intuitive base cases, and a recursion that applies well. The base cases we usually have to consider are of a node-pointer being `NULL` , and of a node's next pointer being `NULL` ; and the recursive case usually does something to the current value, and calls itself on the following nodes.

To deal with all the things in a list, we've previously had constructs like this, to loop over a list:

```
Node curr = l->head;
while (curr != NULL) {
    // ... do something
    curr = curr->next;
}
```

Because we're using two structures, one for the list's head, and one for each list node, the main part of the function we're implementing is usually a call into a helper function:

```
void listRecurse (List l) {
    assert (l != NULL);
    return listRecurseHelper (l->head);
}
```

Note that `listRecurse` is *not* recursive, but `listRecurseHelper` would be.

Here's a simple recursion: find the length of a linked list. If we don't have a node, its length is zero; if we have a node, the length is one plus the length of the rest of the list.

Create a file called `listLengthRec.c` that includes the `list.h` header file. In it, you should implement a recursive version of `listLength` , a function which takes a linked list, and returns the length of the list. It should have this prototype:

```
int listLength (List l);
```

To run Styl-o-matic:

```
$ 1511 stylomatic listLengthRec.c
Looks good!
```

You'll get advice if you need to make changes to your code.

Submit your work with the *give* command, like so:

```
$ give cs1511 wk12_listLengthRec
```

Or, if you are working from home, upload the relevant file(s) to the `wk12_listLengthRec` activity on Give Online.