List Length

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

Linked Lists

For this activity, you will be looking at a linked list.

Download `list.h`, or copy it into your current directory on a CSE system by running

```
$ cp /web/cs1511/17s2/week11/files/list.h .
```

A linked list is another way of representing a collection of data of the same type.

Linked lists are made up of *nodes*, which store one element of the list each, as well as the location (in memory) of the next element.

Nodes can store data of any type, but for this week we will be looking at nodes which store integers, which are defined like this:

```
typedef struct _node *Node;

typedef struct _node {
    int value;
    Node next;
} node;
```

As well as the `node` data structure, we also have a `list` data structure. The `list` data structure is used to keep track of what the first element in the list is, and looks like this:

```
typedef struct _list {
    Node head;
} list;
```

By convention, we call the first element in the list the `head` of the list.

List Length

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

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

You should write your own tests in a separate file; `listLength.c` should *not* contain a `main` .

To run some simple automated tests:

```
$ 1511 autotest listLength
```

To run Styl-o-matic:

```
$ 1511 stylomatic listLength.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 wk11_listLength
```

Or, if you are working from home, upload the relevant file(s) to the `wk11_listLength` activity on Give Online.