Testing a Card ADT

This is a **pair** exercise and must be competed in your **tutorial** or **lab** with your partner.

Heads up! Make sure you've done *The Card ADT* before attempting this exercise.

Having trouble? Try the *Testing a Complex ADT* warmup.

We can fairly easily write *black-box* tests for our newly written Card ADT, as black-box tests only require a well-defined interface. Our ADT's interface does fairly sensible things, so you could write a test to check, for example, the color, number, or suit that you put in is the same as the color, number, or suit that comes out.

You can verify that with our good friend `assert()`. `assert()` allows us to state an *invariant* in our program: something that is always true, and if it is not, our program ceases to exist. Some programmers use `assert` as a primitive error handling mechanism; in this course, that's strictly forbidden.

```
color newColor = RED;
suit newSuit = HEARTS;
char newNumber = 2;
Card c = newCard (newNumber, newColor, newSuit);
assert (c != NULL);
assert (cardNumber (c) == newNumber);
assert (cardColor (c) == newColor);
assert (cardSuit (c) == newSuit);
```

One caveat to be aware of: you cannot test destructors. Once you call a destructor, you've almost definitely called `free()`, and you've promised the memory allocator that you'll never use that memory again.

Go forth, and write tests for the functions in the interface. Think about the edge cases, and the combinations of values that might cause your tests to fail.

To run Styl-o-matic:

```
$ 1511 stylomatic testCard.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 wk09_testCardADT
```

Or, if you are working from home, upload the relevant file(s) to the `wk09_testCardADT` activity on Give Online.