**Resistor Color Duo**

In Practice Mode

Introduction

If you want to build something using a Raspberry Pi, you'll probably use *resistors*. For this exercise, you need to know two things about them:

* Each resistor has a resistance value.
* Resistors are small - so small in fact that if you printed the resistance value on them, it would be hard to read. To get around this problem, manufacturers print color-coded bands onto the resistors to denote their resistance values. Each band has a position and a numeric value. For example, if they printed a brown band (value 1) followed by a green band (value 5), it would translate to the number 15.

In this exercise you are going to create a helpful program so that you don't have to remember the values of the bands. The program will take color names as input and output a two digit number, even if the input is more than two colors!

The band colors are encoded as follows:

* Black: 0
* Brown: 1
* Red: 2
* Orange: 3
* Yellow: 4
* Green: 5
* Blue: 6
* Violet: 7
* Grey: 8
* White: 9

From the example above: brown-green should return 15 brown-green-violet should return 15 too, ignoring the third color.

Getting Started

Make sure you have read the "Guides" section of the [C track](https://exercism.io/my/tracks/c) on the Exercism site. This covers the basic information on setting up the development environment expected by the exercises.

Passing the Tests

Get the first test compiling, linking and passing by following the [three rules of test-driven development](http://butunclebob.com/ArticleS.UncleBob.TheThreeRulesOfTdd).

The included makefile can be used to create and run the tests using the test task.

make test

Create just the functions you need to satisfy any compiler errors and get the test to fail. Then write just enough code to get the test to pass. Once you've done that, move onto the next test.

As you progress through the tests, take the time to refactor your implementation for readability and expressiveness and then go on to the next test.

Try to use standard C99 facilities in preference to writing your own low-level algorithms or facilities by hand.