**Hamming**

In Practice Mode

Introduction

Calculate the Hamming Distance between two DNA strands.

Your body is made up of cells that contain DNA. Those cells regularly wear out and need replacing, which they achieve by dividing into daughter cells. In fact, the average human body experiences about 10 quadrillion cell divisions in a lifetime!

When cells divide, their DNA replicates too. Sometimes during this process mistakes happen and single pieces of DNA get encoded with the incorrect information. If we compare two strands of DNA and count the differences between them we can see how many mistakes occurred. This is known as the "Hamming Distance".

We read DNA using the letters C,A,G and T. Two strands might look like this:

GAGCCTACTAACGGGAT

CATCGTAATGACGGCCT

^ ^ ^ ^ ^ ^^

They have 7 differences, and therefore the Hamming Distance is 7.

The Hamming Distance is useful for lots of things in science, not just biology, so it's a nice phrase to be familiar with :)

Implementation notes

The Hamming distance is only defined for sequences of equal length, so an attempt to calculate it between sequences of different lengths should not work. The general handling of this situation (e.g., raising an exception vs returning a special value) may differ between languages.

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.