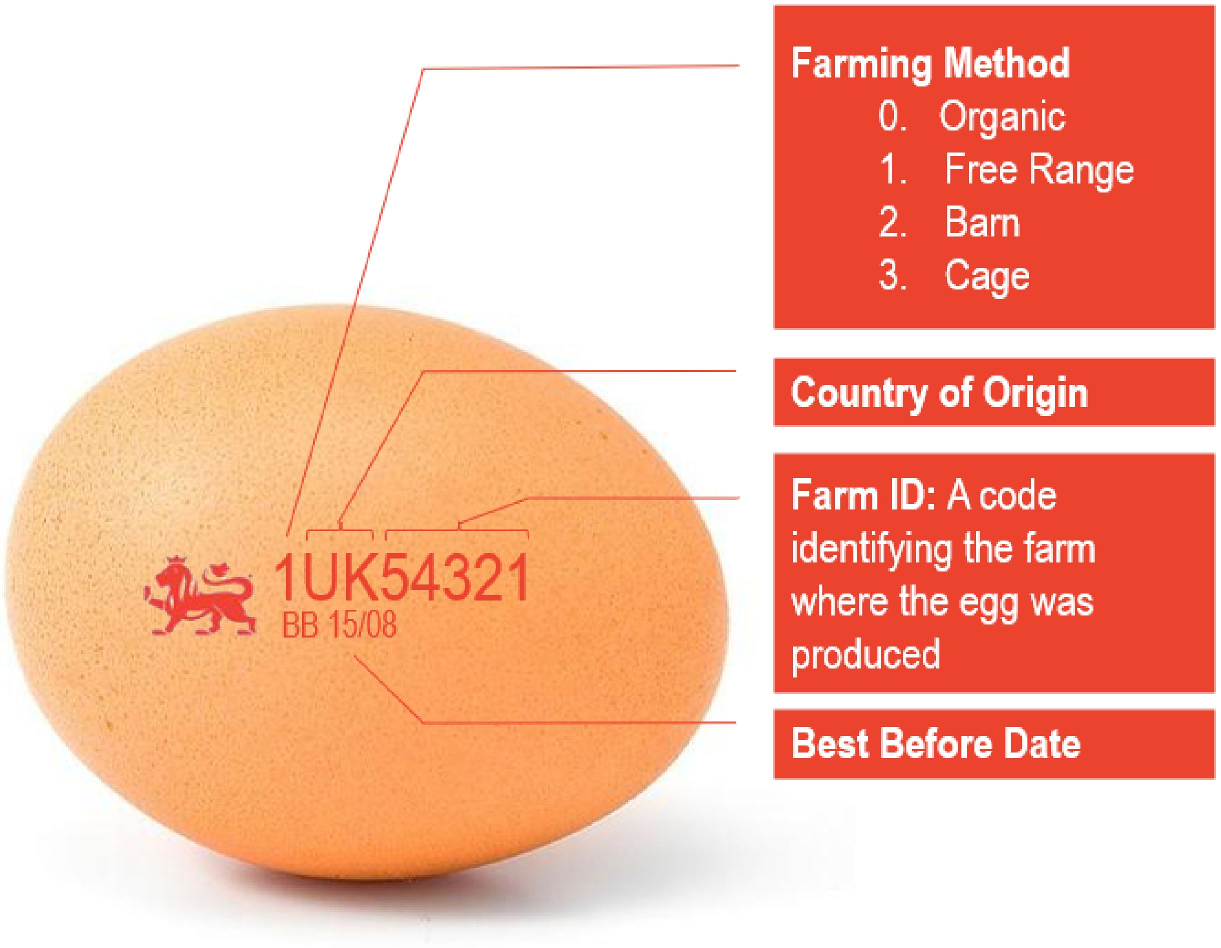
Problem #1
Did you know that in most countries, the eggs you buy in a shop must be stamped with a code that can help you find out more about how and where your eggs were produced. This code is helpful to distinguish organic and free range eggs from eggs from more industrial production (e.g. barn or caged hen eggs).



You will need to write a Python script that:

- Takes one input from the end-user: the code as it appear on a stamped egg. (Just the code, not the Best Before Date)
- Use this code to output the farming method this egg originated from: (Organic, Free range, Barn or Cage)
- Output the country of Origin: e.g.:
    - UK: United Kingdom
    - NL: Netherlands

    - FR: France
    - BE: Belgium
    - DE: Germany
    - ES: Spain
- Output the farm/producer ID

Your program should be able to handle the following situations:

1.Update your code to automatically reject any invalid code:

- A valid code contains at least 7 alphanumeric characters,
- A valid code should start with a number digit between 0 and 3.

2. Our program needs to recognise all the possible two-letter country codes. We have stored the full list of country codes in the following file.

Country code text file:
https://drive.google.com/file/d/1eBcGPjpmsOZeLiGqZ5ed9_SELoQYsuuk/view?usp=sharing

Test Plan

	Input	Expected Output	
#1	1UK54321	Free Range Egg Country of Origin: United Kingdom	

		Farm Id: 54321	
#2	0NL6789	Organic Egg Country of Origin: Netherlands Farm Id: 6789	
#3	3ES02468	Barn Egg Country of Origin: Spain Farm Id: 02468	

Problem #2

Cumulative Elevation Gain Calculator

In mountaineering, running or cycling the cumulative elevation gain (aka total ascent) refers to the sum of every gain in elevation throughout an entire trip.

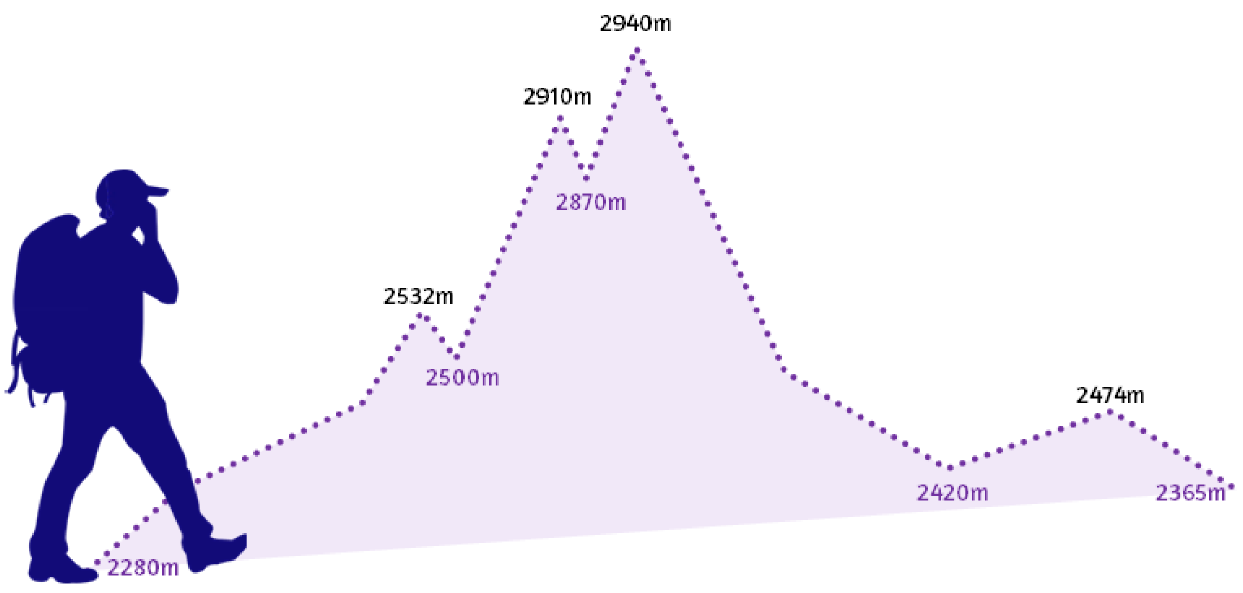
Note that when calculating this elevation gain, elevation losses are not counted in this measure.

When evaluating the difficulty of a hike, you should consider both the total distance of the hike as well as its cumulative elevation gain. Effectively, hiking 10 kilometers on flat land is significantly easier than hiking up and down a series of high peaks over the same distance.

In this challenge, we are going to write a Python script to help a hiker plan their journey by estimating the cumulative elevation gain of their planned hike.

Your Python script will need to:

- Ask the user to enter the altitude of the starting point of the trip,
- For each peak or pit on the planned journey, ask the user to enter the altitude of the peak/or peat,
- Automatically add-up all the corresponding elevation gains (and ignore elevation losses!)
- Output the cumulative elevation gain of the planned hike!



Test Plan

	Input Values	Expected Output	

#1	Starting Altitude: 1800m 1st Peak: 2100m 1st Pit: 1950m 2nd Peak: 2240m Ending Altitude: 1800m	Cumulative Elevation Gain: 590m	
#2	Starting Altitude: 1500m 1st Peak: 1850m 1st Pit: 1800m 2nd Peak: 2100m 2nd Pit: 1900m 3rd Peak: 2400m (End of Trek)	Cumulative Elevation Gain: 1150m	

Problem #3
Proportions and cross products

A proportion is simply a statement that two ratios are equal. The following are examples of proportions:

- 1/4 = 25/100
- 8/24 = 1/3
- 40/60 = 2/3

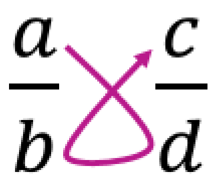
Proportions are often used in Math to simplify a fraction or to represent a fraction as a percentage.

In order two find out if two fractions are equal, we can use one of the following three methods:

- Calculate and compare their decimal values: e.g. 25/100 = 0.25 and 1/4 = 0.25 so 25/100 = 1/4
- Simplify both fractions: e.g. 75/100 = 3/4 and 18/24 = 3/4 so 75/100 = 18/24
- Compare cross products: e.g. 3 x 6 = 2 x 9 so 2/6 = 3/9

Cross Products

You can compare two fractions to find out if they are equal or not if their cross products are equal:



$$\text{if } a \times d = c \times b \text{ then } \frac{a}{b} = \frac{c}{d}$$

Your aim is to write a Python program that will check if two fractions are equal or not using the cross products approach

- . The program will take 4 inputs: the numerator and denominator of both fractions
- .It will then compare the cross products to decide and output if the two fractions are equal or not.

Test Plan

Complete the following tests to check that your code is working as expected:

	Input Values	**Expected Output**	
#1	Fraction #1: 4/6 Fraction #2: 20/30	These fractions are equal.	
#2	Fraction #1: 4/6 Fraction #2: 20/40	These fractions are not equal.	

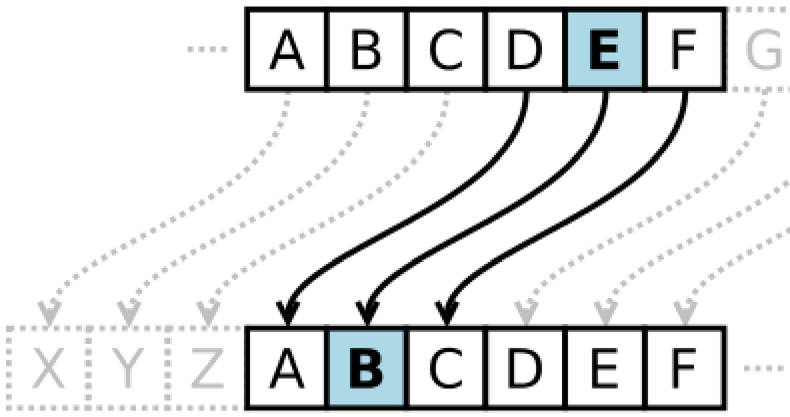
#3	Fraction #1: 75/100 Fraction #2: 3/4	These fractions are equal.	
#4	Fraction #1: 1/3 Fraction #2: 30/100	These fractions are not equal.	

Problem #4

Caesar Cipher

In cryptography, a Caesar cipher, also known as shift cipher, is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. For example, with a left shift of 3, D would be replaced by A, E would become B, and so on. The method is named after Julius Caesar, who used it in his private correspondence.

The action of a Caesar cipher is to replace each plaintext letter with a different one a fixed number of places down the alphabet.



The cipher illustrated above uses a left shift of three, so that each occurrence of E in the plaintext becomes B in the ciphertext.

The transformation can be represented by aligning two alphabets; the cipher alphabet is the plain alphabet rotated left or right by some number of positions. For instance, here is a Caesar cipher using a left rotation of three places, equivalent to a right shift of 23 (the shift parameter is used as the key):

```
Plain:  ABCDEFGHIJKLMNOPQRSTUVWXYZ

Cipher: XYZABCDEFGHIJKLMNOPQRSTUVW
```

When encrypting, a person looks up each letter of the message in the “plain” line and writes down the corresponding letter in the “cipher” line.

```
Plaintext:  THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG

Ciphertext: QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV ALD
```

Deciphering is done in reverse, with a right shift of 3.

The encryption can also be represented using modular arithmetic by first transforming the letters into numbers, according to the scheme, A = 0, B = 1,…, Z = 25.[2] Encryption of a letter x by a shift n can be described mathematically as,

$$E_n(x) = (x + n) \mod 26.$$

Decryption is performed similarly,

$$D_n(x) = (x - n) \mod 26.$$

Write a python script to input a message in plaintext, and then print the encrypted message as output.

Test

```
Plain:  ABCDEFGHIJKLMNOPQRSTUVWXYZ

Cipher: XYZABCDEFGHIJKLMNOPQRSTUVW
```