**Roman numerals**

You've most likely seen Roman numerals, even if you didn't recognize them. You may have seen them in copyrights of old movies and television shows (“Copyright MCMXLVI” instead of “Copyright 1946”), or on the dedication walls of libraries or universities (“established MDCCCLXXXVIII” instead of “established 1888”). You may also have seen them in outlines and bibliographical references. It's a system of representing numbers that really does date back to the ancient Roman empire (hence the name).

In Roman numerals, there are seven characters that are repeated and combined in various ways to represent numbers.

* I = 1
* V = 5
* X = 10
* L = 50
* C = 100
* D = 500
* M = 1000

The following are some general rules for constructing Roman numerals:

* Characters are additive. I is 1, II is 2, and III is 3. VI is 6 (literally, “5 and 1”), VII is 7, and VIII is 8.
* The tens characters (I, X, C, and M) can be repeated up to three times. At 4, you need to subtract from the next highest fives character. You can't represent 4 as IIII; instead, it is represented as IV (“1 less than 5”). The number 40 is written as XL (10 less than 50), 41 as XLI, 42 as XLII, 43 as XLIII, and then 44 as XLIV (10 less than 50, then 1 less than 5).
* Similarly, at 9, you need to subtract from the next highest tens character: 8 is VIII, but 9 is IX (1 less than 10), not VIIII (since the I character can not be repeated four times). The number 90 is XC, 900 is CM.
* The fives characters can not be repeated. The number 10 is always represented as X, never as VV. The number 100 is always C, never LL.
* Roman numerals are always written highest to lowest, and read left to right, so the order the of characters matters very much. DC is 600; CD is a completely different number (400, 100 less than 500). CI is 101; IC is not even a valid Roman numeral (because you can't subtract 1 directly from 100; you would need to write it as XCIX, for 10 less than 100, then 1 less than 10).

**Thousands in Roman Letters:**

What would it take to validate that an arbitrary string is a valid Roman numeral? Let's take it one digit at a time. Since Roman numerals are always written highest to lowest, let's start with the highest: the thousands place. For numbers 1000 and higher, the thousands are represented by a series of M characters.

Ex: 4000 = MMMM

**Hundreds in Roman Letters:**

The hundreds place is more difficult than the thousands, because there are several mutually exclusive ways it could be expressed, depending on its value.

* 100 = C
* 200 = CC
* 300 = CCC
* 400 = CD
* 500 = D
* 600 = DC
* 700 = DCC
* 800 = DCCC
* 900 = CM

So there are four possible patterns:

* CM
* CD
* Zero to three C characters (zero if the hundreds place is 0)
* D, followed by zero to three C characters

The last two patterns can be combined:

* an optional D, followed by zero to three C characters

**Introduction to Roman numerals Testing**

The rules for Roman numerals lead to a number of interesting observations:

1. There is only one correct way to represent a particular number as Roman numerals.

The converse is also true: if a string of characters is a valid Roman numeral, it represents only one number (i.e. it can only be read one way).

2. There is a limited range of numbers that can be expressed as Roman numerals, specifically 1 through 4999. (The Romans did have several ways of expressing larger numbers, for instance by having a bar over a numeral to represent that its normal value should be multiplied by 1000, but you're not going to deal with that. For the purposes of this assignment let's stipulate that Roman numerals go from 1 to 4999.)

3. There is no way to represent 0 in Roman numerals. (Amazingly, the ancient Romans had no concept of 0 as a number.

4. Numbers were for counting things you had; how can you count what you don't have?)

5. There is no way to represent negative numbers in Roman numerals.

6. There is no way to represent fractions or non−integer numbers in Roman numerals.

Given all of this, what would you expect out of a set of functions to convert to and from Roman numerals?

roman.py requirements

1. toRoman should return the Roman numeral representation for all integers 1 to 4999.

2. toRoman should fail when given an integer outside the range 1 to 4999.

3. toRoman should fail when given a non−integer number.

4. fromRoman should take a valid Roman numeral and return the number that it represents.

5. fromRoman should fail when given an invalid Roman numeral.

6. If you take a number, convert it to Roman numerals, then convert that back to a number, you should end up with the number you started with.

So fromRoman(toRoman(n)) == n for all n in 1..4999.

7. toRoman should always return a Roman numeral using uppercase letters.

8. fromRoman should only accept uppercase Roman numerals (i.e. it should fail when given lowercase input).