CS 5035 (Fall 2016)

### Project 3. Syntax in functions (first attempt by Sept 19)

Based on chapter [4 of LYH](http://learnyouahaskell.com/syntax-in-functions). [Videos](https://sites.google.com/a/lclark.edu/drake/courses/pls/lesson-3-haskell-syntax-in-functions).

Write a function that converts numbers expressed as Roman numerals to Arabic (base 10). The code below (modified slightly from [here](https://www.rosettacode.org/wiki/Roman_numerals/Decode#Haskell)) does it cleverly and recursively. (There are a number of more complex solutions on the web.)

Even though we haven’t yet talked about recursion in Haskell, the recursion in toArabic is simple enough that you should be able to understand it. (If not look ahead to the section on recursion.) If you still need help, ask.

**import** Data.List (isPrefixOf)

mapping = [("M",1000),("CM",900),("D",500),("CD",400),("C",100),("XC",90),

("L",50),("XL",40),("X",10),("IX",9),("V",5),("IV",4),("I",1)]

toArabic :: String -> Int

toArabic "" = 0

toArabic str = num + toArabic rest

where (num, rest) = oneStep str

{-

Note: could also be written as follows.

toArabic str =

let (num, rest) = oneStep str

in num + toArabic rest

-}

oneStep :: String -> (Int, String)

oneStep str =

head [(num, drop (length roman) str) |

(roman,num) <- mapping, roman `isPrefixOf` str]

testCases = ["MCMXC", "MMVIII", "MDCLXVI"]

test = zip testCases (map toArabic testCases)

\*Main> test

[("MCMXC",1990),("MMVIII",2008),("MDCLXVI",1666)]

Your assignment: (a) explain the code above and (b) rewrite oneStep so that it doesn’t use mapping, isPrefixOf, or list comprehension. Do all the work using guards or patterns.