CS3200: Programming Languages
Homework 8: more on Haskell

1. Roman numerals are a numeral system that originated in ancient Rome and remained the usual way of writing numbers throughout Europe well into the Late Middle Ages. Numbers in this system are represented by combinations of letters from the Latin alphabet. Modern usage employs seven symbols, each with a fixed integer value:

Symbol	I	V	X	L	C	D	M
Value	1	5	10	50	100	500	1,000

   (Source, plus more examples and history: https://en.wikipedia.org/wiki/Roman_numerals)

   (a) Write a function `convertToRoman` which accepts a number (type Int) and converts it to a Roman numeral representation.

   (b) Write a function `convertToNumber` which accepts a String that is a Roman numeral and converts it to an Int.

   Please be sure to use these exact function names (so that they work for my tests), and include a proper declaration (with types or typeclasses specified) above your function.

2. Define a function `subsequence` that takes two lists and returns the ascending list of indices at which the first list occurs as a subsequence of the second list. If there are multiple solutions, return the one with smallest sum of all indices.

```
subsequence ::  Eq a => [a] -> [a] -> [Int]
subsequence "abcde" "abcbcdef" ⟹ [0, 1, 2, 5, 6]
subsequence [9, 9, 7] [9, 7, 7, 9, 9, 7] ⟹ [0, 3, 5]
subsequence "abc" "caccdcbdca" ⟹ [1, 6, 8]
subsequence "312" "1212313" ⟹ error "subsequence does not exist"
```

3. Write a function:

```
squash :: (a -> a -> b) -> [a] -> [b]
```

   which applies a given function to adjacent elements in a list. For example, squash $f[x_1, x_2, x_3, x_4]$ should equal $[fx_1x_2, fx_2x_3, fx_3x_4]$.