Replacement Coursework 1 (Roman Numerals)

This coursework is worth 10%. It is about translating roman numerals into integers and also about validating roman numerals. The coursework is due on 2 February at 5pm. Make sure the files you submit can be processed by just calling `scala <<filename.scala>>`.

Important: Do not use any mutable data structures in your submission! They are not needed. This excludes the use of `ListBuffer`s, for example. Do not use `return` in your code! It has a different meaning in Scala, than in Java. Do not use `var`! This declares a mutable variable. Make sure the functions you submit are defined on the "top-level" of Scala, not inside a class or object. Also note that the running time will be restricted to a maximum of 360 seconds.

Disclaimer

It should be understood that the work you submit represents your own effort! You have not copied from anyone else. An exception is the Scala code I showed during the lectures or uploaded to KEATS, which you can freely use.

Part 1 (Translation)

Roman numerals are strings consisting of the letters I, V, X, L, C, D, and M. Such strings should be transformed into an internal representation using the datatypes `RomanDigit` and `RomanNumeral`, and then from this internal representation converted into an Integer.

(1) First write a polymorphic function that recursively transforms a list of options into an option of a list. For example, if you have the lists on the left, they should be transformed into the option on the right:

```
List(Some(1), Some(2), Some(3))  ⇒  Some(List(1, 2, 3))
List(Some(1), None, Some(3))     ⇒  None
List()                           ⇒  Some(List())
```

This means the function should produce `None` as soon as a `None` is inside the list. Otherwise it produces a list of all `Some`s. In case the list is empty, it produces `Some` of the empty list. [1 Mark]

(2) Write a function first a function that converts a character I, V, X, L, C, D, or M into an option of a `RomanDigit`. If it is one of the roman digits, it should produce `Some`; otherwise `None`.

Next write a function that converts a string into a `RomanNumeral`. Again, this function should return an `Option`: If the string consists of I, V, X, L,

C, D, and M only, then it produces `Some`; otherwise if there is any other character in the string, it should produce `None`. The empty string is just the empty `RomanNumeral`, that is empty list of `RomanDigit`'s. You should use the function under Task (1) to produce the result. [2 Marks]

(3) Write a recursive function RomanNumral2Int that converts a `RomanNumeral` into an integer. You can assume the generated integer will be between 0 and 3999. The argument of the function is a list of roman digits. It should look how this list starts and then calculate what the corresponding integer is for this "start" and add it with the integer for the rest of the list. That means if the argument is of the form shown on the left-hand side, it should do the calculation on the right-hand side.

$$
\begin{array}{lcl}
M :: r & \Rightarrow & 1000 + \text{roman numeral of rest } r \\
C :: M :: r & \Rightarrow & 900 + \text{roman numeral of rest } r \\
D :: r & \Rightarrow & 500 + \text{roman numeral of rest } r \\
C :: D :: r & \Rightarrow & 400 + \text{roman numeral of rest } r \\
C :: r & \Rightarrow & 100 + \text{roman numeral of rest } r \\
X :: C :: r & \Rightarrow & 90 + \text{roman numeral of rest } r \\
L :: r & \Rightarrow & 50 + \text{roman numeral of rest } r \\
X :: L :: r & \Rightarrow & 40 + \text{roman numeral of rest } r \\
X :: r & \Rightarrow & 10 + \text{roman numeral of rest } r \\
I :: X :: r & \Rightarrow & 9 + \text{roman numeral of rest } r \\
V :: r & \Rightarrow & 5 + \text{roman numeral of rest } r \\
I :: V :: r & \Rightarrow & 4 + \text{roman numeral of rest } r \\
I :: r & \Rightarrow & 1 + \text{roman numeral of rest } r
\end{array}
$$

The empty list will be converted into integer 0. [1 Mark]

(4) Write a function that takes a string and if possible converts it into the internal representation. If successful, then calculate the integer (an option of an integer) according to the function in (3). If this is not possible, then return `None`. [1 Mark]

(5) The file `roman.txt` contains a list of roman numerals. Read in these numerals, convert them into integers and then add them all up. The function for reading a file is

```
Source.fromFile("filename")("ISO-8859-9")
```

Make sure you process the strings correctly by ignoring whitespaces where neded.

[1 Mark]

Part 2 (Validation)