Worksheet 2: *Lists*

Template file:	Worksheet2.hs
Labs:	5 January, 2021
Hand-in: Worksheet2.hs	12 February, 2021at 12:00hr
Topics:	Lists. Map and filter. List comprehension.

Message: (1) don't forget to put your name on your script, (2) scripts that don't compile properly may loose 20% points, (so comment out the lines that don't work, and add if you wish some explanation) (3) Take care that the layout is pleasing and your answers are easy to read (4) More importantly have fun!

1. A Bank stores details on its customers via their national insurance number, their age, and their balance. This gives the following type definitions.

```
type NI = Int
type Age = Int
type Balance = Float
type Customer  = (NI,Age, Balance)
type Bank = [Customer]
```

   (a) (10 marks) Define a function `retired :: Customer -> Bool` which returns true if the person is, or is over, 67 years.

   (b) (10 marks) Define a function `deposit :: Customer -> Float -> Customer` which adds a given amount to the person's balance.

   (c) (10 marks) Define a function `withdraw :: Customer -> Float -> Customer` which removes a given amount from the person's balance, but only if the remaining total is positive!

   (d) (10 marks) Define a function `credit :: Bank -> [Customer]` which returns those people who are not overdrawn.

2. (10 marks) Define a function `addIndex :: [Int] -> [(Int,Int)]` that given a list $[n_1, n_2, \ldots n_k]$ of integers produces the list $[(1, n_1), (2, n_2), \ldots, (k, n_k)]$ which is a list of pairs of integers.

   For example `addIndex [2,2,3,1] -> [(1,2),(2,2),(3,3),(4,1)]`

3. (10 marks)

   Define a function `reproduce :: Int -> String -> [String]` that given a number n and a string outputs a list with n replicates of the given string.

   Use list comprehension. Hint: remember the previous exercise.

4. (30 Marks) Write down a function `encode :: Int -> String -> String` that given an integer n and a string *text* shifts all the letters of the alphabet n places (if you reach z, start with a again). Lower (upper) case letters should remain lower (upper) case. All other symbols should remain the same (not a particularly clever way of coding...)

   Example `encode 1 "abcdxyz" = "bcdeyza"` and `encode 2 "abcXYZ" = "cdeZAB"`.

   Hint: chr and ord may be useful. First make a function `code :: Int -> Char -> Char`.

5. (10 Marks) Read the section on Escaping text on `http://book.realworldhaskell.org/read/characters-strings-and-escaping-rules.html`.

   Now write down a string `rawtext` such that `putStr rawtext` outputs exactly

```
"This is a \ \long string,
\ \ spanning multiple lines,
in fact 3 lines!"
```