# Square Code (Scheme+Haskell - 7+7 Points)

August 24, 2022

One classic method for composing secret messages is called a *square code*. First, the input is normalized: the spaces and punctuation are removed from the English text, and the message is downcased. Then, the normalized characters are broken into rows. These rows can be regarded as forming a rectangle. For example, the sentence

```
If man was meant to stay on the ground, god would have given us roots.
```

is normalized to:

```
ifmanwasmeanttostayonthegroundgodwouldhavegivenusroots
```

The normalized string is 54 characters long, so it is written into a rectangle with 7 rows and 8 columns.

```
ifmanwas
meanttos
tayonthe
groundgo
dwouldha
vegivenu
sroots
```

Note that the last row is padded with spaces at the end to make it 8 characters long.
The coded message is obtained by reading down the columns going left to right. For example, the message above is coded as:

```
imtgdvs fearwer mayoogo anouuio ntnnlvt wttddes aohghn  sseoau
```

Given the length $n \in \mathbb{N}$ of the normalized text, the number of columns $c \in \mathbb{N}$ in the rectangle is computed by $c = \lceil \sqrt{n} \rceil$, i.e., $c$ is the smallest natural number greater than or equal to $\sqrt{n}$.

## Task 3 - Scheme

In Scheme, implement a function `(encode str)` that accepts a string and returns the encoded string as described above. Your file is to be called `task3.rkt` and must provide the function `encode`. Hence, the head of your file should start with

```
#lang racket
(provide encode)

; your code here
```

### Hint

To make the string downcased, you may use the function `(string-downcase str)`. To remove spaces and punctuation, you may use the predicate `(char-alphabetic? ch)`. To get the least integer above a real number $x$, use the function `(exact-ceiling x)`.

### Examples

The following shows the behaviour of the `encode` function.

```
> (encode "If man was meant to stay on the ground, god would have given us roots.")
"imtgdvs fearwer mayoogo anouuio ntnnlvt wttddes aohghn  sseoau "
```

```
> (encode "Have a nice day!")
"hae and via ecy"
```

## Task 4 - Haskell

In Haskell, implement a function `encode :: String -> String` that accepts a string and returns the encoded string as described above. Your task is to be called `Task4.rkt` and must export the `encode` function. Hence, the head of your file should read

```
module Task4 ( encode ) where
import Data.Char

-- your code goes here
```

### Hint

To make the string downcased, you may use the function `toLower :: Char -> Char`. To remove spaces and punctuation, you may use the predicate `isAlpha :: Char -> Bool`. Both functions are located at the module `Data.Char`. To get the least integer above a real number, use the function `ceiling :: Integral b => a -> b`.

### Examples

The following shows the behaviour of the `encode` function.

```
> encode "If man was meant to stay on the ground, god would have given us roots."
"imtgdvs fearwer mayoogo anouuio ntnnlvt wttddes aohghn  sseoau "
```

```
> encode "haveaniceday"
"hae and via ecy"
```