Crypto Square in Scala

Readme (readme) | Test Suite (../crypto-square)

Crypto Square

Implement the classic method for composing secret messages called a square code.

The input is first 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 when printed with intervening newlines.

For example, the sentence

> If man was meant to stay on the ground god would have given us roots

is 54 characters long.

Broken into 8-character columns, it yields 7 rows.

Those 7 rows produce this rectangle when printed one per line:

```
1 ifmanwas
2 meanttos
3 tayonthe
4 groundgo
5 dwouldha
6 vegivenu
7 sroots
```

The coded message is obtained by reading down the columns going left to right.

For example, the message above is coded as:

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

Write a program that, given an English text, outputs the encoded version of that text.

The size of the square (number of columns) should be decided by the length of the message.

If the message is a length that creates a perfect square (e.g. 4, 9, 16, 25, 36, etc), use that number of columns.

If the message doesn't fit neatly into a square, choose the number of columns that corresponds to the smallest square that is larger than the number of characters in the message.

For example, a message 4 characters long should use a 2 x 2 square. A message that is 81 characters long would use a square that is 9 colums wide.

A message between 5 and 8 characters long should use a rectangle 3 characters wide.

Output the encoded text grouped by column.

For example:

- "Have a nice day. Feed the dog & chill out!"
    - Normalizes to: "haveanicedayfeedthedogchillout"
    - Which has length: 30
    - And splits into 5 6-character rows:
    - "havean"
    - "iceday"
    - "feedth"
    - "edogch"
    - "illout"
    - Which yields a ciphertext beginning: "hifei acedl v…"

The Scala exercises assume an SBT project scheme. The exercise solution source should be placed within the exercise directory/src/main/scala. The exercise unit tests can be found within the exercise directory/src/test/scala.

To run the tests simply run the command `sbt test` in the exercise directory.

For more detailed info about the Scala track see the help page (http://help.exercism.io/getting-started-with-scala.html).

Source

J Dalbey's Programming Practice problems view source (http://users.csc.calpoly.edu/%7Ejdalbey/103/Projects/ProgrammingPractice.html)

exercism.io
(/)
Beta

About (/about) - Donate (/donate)

GitHub (https://github.com/exercism/exercism.io) Twitter (https://twitter.com/exercism_io)

Newsletter (https://tinyletter.com/exercism)

SPONSORS