Crosswords-101 ★

Points: 553.1800000000001 Rank: 1150

Problem Submissions Leaderboard

RATE THIS CHALLENGE

★ ★ ★ ★ ★

A 10×10 Crossword grid is provided to you, along with a set of words (or names of places) which need to be filled into the grid.

The cells in the grid are initially, either + signs or − signs.

Cells marked with a + have to be left as they are. Cells marked with a − need to be filled up with an appropriate character.

Input Format

The input contains 10 lines, each with 10 characters (which will be either + or - signs).

After this follows a set of words (typically nouns and names of places), separated by semi-colons (;).

Constraints

There will be no more than ten words. Words will only be composed of upper-case `A-Z` characters. There will be no punctuation (hyphen, dot, etc.) in the words.

Output Format

Position the words appropriately in the 10×10 grid, and then display the 10×10 grid as the output. So, your output will consist of 10 lines with 10 characters each.

Sample Input 0

```
+-++++++++
+-++++++++
+-++++++++
+-----++++
+-+++-++++
+-+++-++++
+++++-++++
++------++
+++++-++++
+++++-++++
LONDON;DELHI;ICELAND;ANKARA
```

Sample Output 0

```
+L++++++++
+O++++++++
+N++++++++
+DELHI++++
+O+++C++++
+N+++E++++
+++++L++++
++ANKARA++
+++++N++++
+++++D++++
```

Sample Input 1

```
+-++++++++
+-++++++++
+-------++
```

```
+-++++++++
+-++++++++
+------+++
+-+++-++++
+++++-++++
+++++-++++
++++++++++
AGRA;NORWAY;ENGLAND;GWALIOR
```

Sample Output 1

```
+E++++++++
+N++++++++
+GWALIOR++
+L++++++++
+A++++++++
+NORWAY+++
+D+++G++++
+++++R++++
+++++A++++
++++++++++
```

Change Theme Language Haskell

```haskell
{-# LANGUAGE TupleSections #-}

import Data.List as L
import Data.Map as M
import Data.Ord (comparing)
import Data.Maybe
import Control.Monad


getSlots :: [String] -> [[(Int, Int)]]
getSlots grid = L.filter ((> 1) . length) $ scanLines 0 grid ++ fmap (fmap exchange) (scanLines 0 (L.transpose grid))
    where scanLines _ [] = []
          scanLines i (l:ls) = scanLine i 0 l ++ scanLines (i+1) ls
          scanLine _ _ ""  = []
          scanLine i j str = let (s, ss) = span (== '+') str
                                 jj = j + length s
                                 (s', ss') = span (== '-') ss
                                 jjj = jj + length s' - 1
                             in fmap (i,) [jj..jjj] : scanLine i (jjj+1) ss'
          exchange (x,y) = (y,x)

findResult :: [[(Int, Int)]] -> [String] -> [([(Int, Int)], String)]
findResult slots ws = fromJust $  L.find g $ doFind (f slots) (f ws)
    where f = L.groupBy (\x y -> length x == length y) . L.sortBy (comparing length)
          g result = isJust $ L.foldl' (\mm (p, ch) ->
              case mm of
                Just m -> let ch' = M.lookup p m
```

Line: 50 Col: 1

Test against custom input

Upload Code as File

Run Code Submit Code

Upload Code as File

Run Code Submit Code