99 questions/95 to 99

From HaskellWiki

< 99 questions

This is part of Ninety-Nine Haskell Problems, based on Ninety-Nine Prolog Problems (https://prof.ti.bfh.ch/hew1/informatik3/prolog/p-99/) .

1 Miscellaneous problems

2 Problem 95

(**) English number words

On financial documents, like cheques, numbers must sometimes be written in full words. Example: 175 must be written as one-seven-five. Write a predicate full-words/1 to print (non-negative) integer numbers in full words.

Example in Haskell:

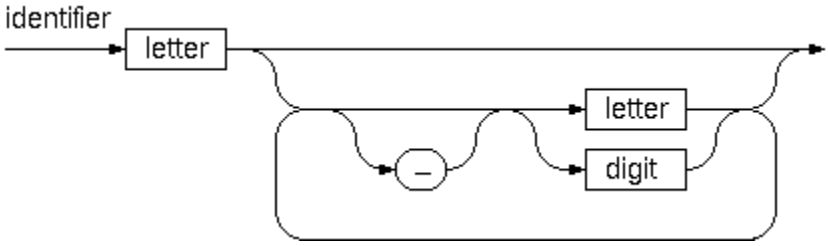
```
> fullWords 175
one-seven-five
```

Solutions

3 Problem 96

(**) Syntax checker

In a certain programming language (Ada) identifiers are defined by the syntax diagram below.



Transform the syntax diagram into a system of syntax diagrams which do not contain loops; i.e. which are purely recursive. Using these modified diagrams, write a predicate identifier/1 that can check whether or not a given string is a legal identifier.

Example in Prolog:

```
% identifier(Str) :- Str is a legal identifier
```

Example in Haskell:

```
> identifier "this-is-a-long-identifier"
True
> identifier "this-ends-in-"
False
> identifier "two--hyphens"
False
```

Solutions

4 Problem 97

(**) Sudoku

Sudoku puzzles go like this:

```
      Problem statement                Solution

       .  .  4 | 8  .  . | .  1  7         9  3  4 | 8  2  5 | 6  1  7
               |         |                         |         |
       6  7  . | 9  .  . | .  .  .         6  7  2 | 9  1  4 | 8  5  3
               |         |                         |         |
       5  .  8 | .  3  . | .  .  4         5  1  8 | 6  3  7 | 9  2  4
       --------+---------+--------         --------+---------+--------
       3  .  . | 7  4  . | 1  .  .         3  2  5 | 7  4  8 | 1  6  9
               |         |                         |         |
       .  6  9 | .  .  . | 7  8  .         4  6  9 | 1  5  3 | 7  8  2
               |         |                         |         |
       .  .  1 | .  6  9 | .  .  5         7  8  1 | 2  6  9 | 4  3  5
       --------+---------+--------         --------+---------+--------
       1  .  . | .  8  . | 3  .  6         1  9  7 | 5  8  2 | 3  4  6
               |         |                         |         |
       .  .  . | .  .  6 | .  9  1         8  5  3 | 4  7  6 | 2  9  1
               |         |                         |         |
       2  4  . | .  .  1 | 5  .  .         2  4  6 | 3  9  1 | 5  7  8
```

Every spot in the puzzle belongs to a (horizontal) row and a (vertical) column, as well as to one single 3x3 square (which we call "square" for short). At the beginning, some of the spots carry a single-digit number between 1 and 9. The

problem is to fill the missing spots with digits in such a way that every number between 1 and 9 appears exactly once in each row, in each column, and in each square.

Solutions

5 Problem 98

(***) Nonograms

Around 1994, a certain kind of puzzle was very popular in England. The "Sunday Telegraph" newspaper wrote: "Nonograms are puzzles from Japan and are currently published each week only in The Sunday Telegraph. Simply use your logic and skill to complete the grid and reveal a picture or diagram." As a Prolog programmer, you are in a better situation: you can have your computer do the work! Just write a little program ;-).

The puzzle goes like this: Essentially, each row and column of a rectangular bitmap is annotated with the respective lengths of its distinct strings of occupied cells. The person who solves the puzzle must complete the bitmap given only these lengths.

```
           Problem statement:          Solution:
```

```
           |_|_|_|_|_|_|_|_| 3         |_|X|X|X|_|_|_|_| 3
           |_|_|_|_|_|_|_|_| 2 1       |X|X|_|X|_|_|_|_| 2 1
           |_|_|_|_|_|_|_|_| 3 2       |_|X|X|X|_|_|X|X| 3 2
           |_|_|_|_|_|_|_|_| 2 2       |_|_|X|X|_|_|X|X| 2 2
           |_|_|_|_|_|_|_|_| 6         |_|_|X|X|X|X|X|X| 6
           |_|_|_|_|_|_|_|_| 1 5       |X|_|X|X|X|X|X|_| 1 5
           |_|_|_|_|_|_|_|_| 6         |X|X|X|X|X|X|_|_| 6
           |_|_|_|_|_|_|_|_| 1         |_|_|_|_|X|_|_|_| 1
           |_|_|_|_|_|_|_|_| 2         |_|_|_|X|X|_|_|_| 2
            1 3 1 7 5 3 4 3            1 3 1 7 5 3 4 3
            2 1 5 1                    2 1 5 1
```

For the example above, the problem can be stated as the two lists [[3],[2,1],[3,2],[2,2],[6],[1,5],[6],[1],[2]] and [[1,2],[3,1],[1,5],[7,1],[5],[3],[4],[3]] which give the "solid" lengths of the rows and columns, top-to-bottom and left-to-right, respectively. Published puzzles are larger than this example, e.g. 25 x 20, and apparently always have unique solutions.

Example in Haskell:

```
Nonogram> putStr $ nonogram [[3],[2,1],[3,2],[2,2],[6],[1,5],[6],[1],[2]] [[1,2],[3,1],[1,5],
|_|X|X|X|_|_|_|_| 3
```

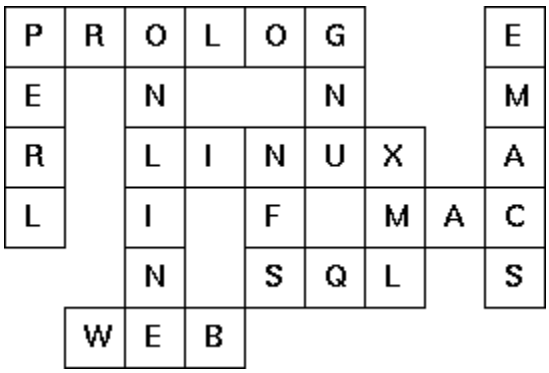
```
|X|X|_|X|_|_|_|_| 2 1
|_|X|X|X|_|_|X|X| 3 2
|_|_|X|X|_|_|X|X| 2 2
|_|_|X|X|X|X|X|X| 6
|X|_|X|X|X|X|X|_| 1 5
|X|X|X|X|X|X|_|_| 6
|_|_|_|_|_|X|_|_| 1
|_|_|_|_|X|X|_|_| 2
 1 3 1 7 5 3 4 3
 2 1 5 1
```

Solutions

6 Problem 99

(***) Crossword puzzle

Given an empty (or almost empty) framework of a crossword puzzle and a set of words. The problem is to place the words into the framework.



The particular crossword puzzle is specified in a text file which first lists the words (one word per line) in an arbitrary order. Then, after an empty line, the crossword framework is defined. In this framework specification, an empty character location is represented by a dot (.). In order to make the solution easier, character locations can also contain predefined character values. The puzzle above is defined in the file p99a.dat (http://www.hta-bi.bfh.ch/~hew/informatik3/prolog/p-99/p99a.dat) , other examples are p99b.dat (http://www.hta-bi.bfh.ch/~hew/informatik3/prolog/p-99/p99b.dat) and p99d.dat (http://www.hta-bi.bfh.ch/~hew/informatik3/prolog/p-99/p99d.dat) . There is also an example of a puzzle (p99c.dat (http://www.hta-bi.bfh.ch/~hew/informatik3/prolog/p-99/p99c.dat)) which does not have a solution.

Words are strings (character lists) of at least two characters. A horizontal or vertical sequence of character places in the crossword puzzle framework is called a site. Our problem is to find a compatible way of placing words onto sites.

Hints: (1) The problem is not easy. You will need some time to thoroughly understand it. So, don't give up too early! And remember that the objective is a clean solution, not just a quick-and-dirty hack!

(2) Reading the data file is a tricky problem for which a solution is provided in the file p99-readfile.pl (http://www.hta-bi.bfh.ch/~hew/informatik3/prolog/p-99/p99-readfile.pl) . See the predicate read_lines/2.

(3) For efficiency reasons it is important, at least for larger puzzles, to sort the words and the sites in a particular order. For this part of the problem, the solution of P28 may be very helpful.

Example in Haskell:

```
ALPHA
ARES
POPPY

  .
  .
.....
  . .
  . .
    .
```

```
> solve $ readCrossword "ALPHA\nARES\nPOPPY\n\n  .  \n  .  \n.....\n  . .\n  . .\n    .\n"

[[((3,1),'A'),((3,2),'L'),((3,3),'P'),((3,4),'H'),((3,5),'A'),((1,3),'P'),((2,3)
,'O'),((3,3),'P'),((4,3),'P'),((5,3),'Y'),((3,5),'A'),((4,5),'R'),((5,5),'E'),((
6,5),'S')]]
```

Solutions

Retrieved from "https://wiki.haskell.org/index.php?title=99_questions/95_to_99&oldid=36225"
Category:

- Tutorials
