Ninety-Nine Lisp Problems

Alexander Burger
abu@software-lab.de

2011-12-01

Ninety-Nine Lisp Problems

Based on a Prolog problem list by werner.hett@hti.bfh.ch. The original is HERE[1].

Work in progress! Until now, only about half of the problems are solved. Another possibility, of course, would be translating the Prolog solutions to Pilog ;-)

Working with lists

P01[2] (*) Find the last box of a list.

```
: (my-last '(a b c d))
-> (d)
```

P02[3] (*) Find the last but one box of a list.

```
: (my-but-last '(a b c d))
-> (c d)
```

P03[4] (*) Find the K'th element of a list.

The first element in the list is number 1.

```
: (element-at '(a b c d e) 3)
-> c
```

[1] https://prof.ti.bfh.ch/hew1/informatik3/prolog/p-99
[2] http://picolisp.com/5000/!wiki?99p01
[3] http://picolisp.com/5000/!wiki?99p02
[4] http://picolisp.com/5000/!wiki?99p03

P04[5] (*) Find the number of elements of a list.

P05[6] (*) Reverse a list.

P06[7] (*) Find out whether a list is a palindrome.

A palindrome can be read forward or backward; e.g. (x a m a x).

P07[8] (**) Flatten a nested list structure.

Transform a list, possibly holding lists as elements into a 'flat' list by replacing each list with its elements (recursively).

```
: (my-flatten '(a (b (c d) e)))
-> (a b c d e)
```

P08[9] (**) Eliminate consecutive duplicates of list elements.

If a list contains repeated elements they should be replaced with a single copy of the element. The order of the elements should not be changed.

```
: (compress '(a a a a b c c a a d e e e e))
-> (a b c a d e)
```

P09[10] (**) Pack consecutive duplicates of list elements into sublists.

If a list contains repeated elements they should be placed in separate sublists.

```
: (consecDups '(a a a a b c c a a d e e e e))
-> ((a a a a) (b) (c c) (a a) (d) (e e e e))
```

P10[11] (*) Run-length encoding of a list.

Use the result of problem P09 to implement the so-called run-length encoding data compression method. Consecutive duplicates of elements are encoded as lists (N E) where N is the number of duplicates of the element E.

```
: (encode '(a a a a b c c a a d e e e e))
-> ((4 a) (1 b) (2 c) (2 a) (1 d)(4 e))
```

P11[12] (*) Modified run-length encoding.

Modify the result of problem P10 in such a way that if an element has no duplicates it is simply copied into the result list. Only elements with duplicates are transferred as (N E) lists.

[5] http://picolisp.com/5000/!wiki?99p04
[6] http://picolisp.com/5000/!wiki?99p05
[7] http://picolisp.com/5000/!wiki?99p06
[8] http://picolisp.com/5000/!wiki?99p07
[9] http://picolisp.com/5000/!wiki?99p08
[10] http://picolisp.com/5000/!wiki?99p09
[11] http://picolisp.com/5000/!wiki?99p10
[12] http://picolisp.com/5000/!wiki?99p11

```
: (encode-modified '(a a a a b c c a a d e e e e))
-> ((4 a) b (2 c) (2 a) d (4 e))
```

P12[13] (**) Decode a run-length encoded list.

Given a run-length code list generated as specified in problem P11. Construct its uncompressed version.

```
: (decode '((4 a) b (2 c) (2 a) d (4 e)))
-> (a a a a b c c a a d e e e e)
```

P13[14] (**) Run-length encoding of a list (direct solution).

Implement the so-called run-length encoding data compression method directly. I.e. don't explicitly create the sublists containing the duplicates, as in problem P09, but only count them. As in problem P11, simplify the result list by replacing the singleton lists (1 X) by X.

```
: (encode-direct '(a a a a b c c a a d e e e e))
-> ((4 a) b (2 c) (2 a) d (4 e))
```

P14[15] (*) Duplicate the elements of a list.

```
: (dupli '(a b c c d))
-> (a a b b c c c c d d)
```

P15[16] (**) Replicate the elements of a list a given number of times.

```
: (repli '(a b c) 3)
-> (a a a b b b c c c)
```

P16[17] (**) Drop every N'th element from a list.

```
: (drop '(a b c d e f g h i k) 3)
-> (a b d e g h k)
```

P17[18] (*) Split a list into two parts; the length of the first part is given.

Do not use any predefined predicates.

```
: (splitAt '(a b c d e f g h i k) 3)
-> ((a b c) (d e f g h i k))
```

[13] http://picolisp.com/5000/!wiki?99p12
[14] http://picolisp.com/5000/!wiki?99p13
[15] http://picolisp.com/5000/!wiki?99p14
[16] http://picolisp.com/5000/!wiki?99p15
[17] http://picolisp.com/5000/!wiki?99p16
[18] http://picolisp.com/5000/!wiki?99p17

P18[19] (**) Extract a slice from a list.

Given two indices, I and K, the slice is the list containing the elements between the I'th and K'th element of the original list (both limits included). Start counting the elements with 1.

```
: (slice '(a b c d e f g h i k) 3 7)
-> (c d e f g)
```

P19[20] (**) Rotate a list N places to the left.

```
: (rotate '(a b c d e f g h) 3)
-> (d e f g h a b c)

: (rotate '(a b c d e f g h) -2)
-> (g h a b c d e f)
```

Hint: Use the predefined functions length and append, as well as the result of problem P17.

P20[21] (*) Remove the K'th element from a list.

```
: (remove-at '(a b c d) 2)
-> (a c d)
```

P21[22] (*) Insert an element at a given position into a list.

```
: (insert-at 'alfa '(a b c d) 2)
-> (a alfa b c d)
```

P22[23] (*) Create a list containing all integers within a given range.

If first argument is smaller than second, produce a list in decreasing order.

```
: (range 4 9)
-> (4 5 6 7 8 9)
```

P23[24] (**) Extract a given number of randomly selected elements from a list.

The selected items shall be returned in a list.

```
: (rnd-select '(a b c d e f g h) 3)
-> (e d a)
```

Hint: Use the built-in random number generator and the result of problem P20.

[19] http://picolisp.com/5000/!wiki?99p18
[20] http://picolisp.com/5000/!wiki?99p19
[21] http://picolisp.com/5000/!wiki?99p20
[22] http://picolisp.com/5000/!wiki?99p21
[23] http://picolisp.com/5000/!wiki?99p22
[24] http://picolisp.com/5000/!wiki?99p23

P24[25] (*) Lotto: Draw N different random numbers from the set 1..M.

The selected numbers shall be returned in a list.

```
: (lotto-select 6 49)
-> (23 1 17 33 21 37)
```

Hint: Combine the solutions of problems P22 and P23.

P25[26] (*) Generate a random permutation of the elements of a list.

```
: (rnd-permu '(a b c d e f))
-> (b a d c e f)
```

Hint: Use the solution of problem P23.

P26[27] (**) Generate the combinations of K distinct objects chosen from the N elements of a list

In how many ways can a committee of 3 be chosen from a group of 12 people? We all know that there are C(12,3) = 220 possibilities (C(N,K) denotes the well-known binomial coefficients). For pure mathematicians, this result may be great. But *we* want to really generate all the possibilities in a list.

```
: (combination 3 '(a b c d e f))
-> ((a b c) (a b d) (a b e) ... )
```

P27[28] (**) Group the elements of a set into disjoint subsets.

a) In how many ways can a group of 9 people work in 3 disjoint subgroups of 2, 3 and 4 persons? Write a function that generates all the possibilities and returns them in a list.

```
: (group3 '(aldo beat carla david evi flip gary hugo ida))
-> (((aldo beat) (carla david evi) (flip gary hugo ida))
... )
```

b) Generalize the above predicate in a way that we can specify a list of group sizes and the predicate will return a list of groups.

```
: (subsets '(aldo beat carla david evi flip gary hugo ida) '(2 2 5))
-> (((aldo beat) (carla david) (evi flip gary hugo ida))
... )
```

Note that we do not want permutations of the group members; i.e. ((aldo beat) ...) is the same solution as ((beat aldo) ...). However, we make a difference between ((aldo beat) (carla david) ...) and ((carla david) (aldo beat) ...).

You may find more about this combinatorial problem in a good book on discrete mathematics under the term "multinomial coefficients".

[25] http://picolisp.com/5000/!wiki?99p24
[26] http://picolisp.com/5000/!wiki?99p25
[27] http://picolisp.com/5000/!wiki?99p26
[28] http://picolisp.com/5000/!wiki?99p27

P28[29] (**) Sorting a list of lists according to length of sublists

a) We suppose that a list contains elements that are lists themselves. The objective is to sort the elements of this list according to their **length**. E.g. short lists first, longer lists later, or vice versa.

```
: (lsort '((a b c) (d e) (f g h) (d e) (i j k l) (m n) (o)))
-> ((o) (d e) (d e) (m n) (a b c) (f g h) (i j k l))
```

b) Again, we suppose that a list contains elements that are lists themselves. But this time the objective is to sort the elements of this list according to their **length frequency**; i.e., in the default, where sorting is done ascendingly, lists with rare lengths are placed first, others with a more frequent length come later.

```
: (lfsort '((a b c) (d e) (f g h) (d e) (i j k l) (m n) (o)))
-> ((i j k l) (o) (a b c) (f g h) (d e) (d e) (m n))
```

Note that in the above example, the first two lists in the result have length 4 and 1, both lengths appear just once. The third and forth list have length 3 which appears twice (there are two list of this length). And finally, the last three lists have length 2. This is the most frequent length.

P29[30] No task defined yet

P30[31] No task defined yet

Arithmetic

P31[32] (**) Determine whether a given integer number is prime.

```
: (is-prime 7)
-> T
```

P32[33] (**) Determine the greatest common divisor of two positive integer numbers.

Use Euclid's algorithm.

```
: (gcd 36 63)
-> 9
```

P33[34] (*) Determine whether two positive integer numbers are coprime.

Two numbers are coprime if their greatest common divisor equals 1.

```
: (coprime 35 64)
-> T
```

[29] http://picolisp.com/5000/!wiki?99p28
[30] http://picolisp.com/5000/!wiki?99p29
[31] http://picolisp.com/5000/!wiki?99p30
[32] http://picolisp.com/5000/!wiki?99p31
[33] http://picolisp.com/5000/!wiki?99p32
[34] http://picolisp.com/5000/!wiki?99p33

P34[35] (**) Calculate Euler's totient function phi(m).

Euler's so-called totient function phi(m) is defined as the number of positive integers r (1 <= r < m) that are coprime to m.

Example: m = 10: r = 1,3,7,9; thus phi(m) = 4. Note the special case: phi(1) = 1.

```
: (totient-phi 10)
-> 4
```

Find out what the value of phi(m) is if m is a prime number. Euler's totient function plays an important role in one of the most widely used public key cryptography methods (RSA). In this exercise you should use the most primitive method to calculate this function (there are smarter ways that we shall discuss later).

P35[36] (**) Determine the prime factors of a given positive integer.

Construct a flat list containing the prime factors in ascending order.

```
: (prime-factors 315)
-> (3 3 5 7)
```

P36[37] (**) Determine the prime factors of a given positive integer (2).

Construct a list containing the prime factors and their multiplicity.

```
: (prime-factors-mult 315)
-> ((3 2) (5 1) (7 1))
```

Hint: The problem is similar to problem P13.

P37[38] (**) Calculate Euler's totient function phi(m) (improved).

See problem P34 for the definition of Euler's totient function. If the list of the prime factors of a number m is known in the form of problem P36 then the function phi(m) can be efficiently calculated as follows:

Let ((p1 m1) (p2 m2) (p3 m3) ...) be the list of prime factors (and their multiplicities) of a given number m. Then phi(m) can be calculated with the following formula:

```
phi(m) = (p1 - 1) * p1 ** (m1 - 1) + (p2 - 1) * p2 ** (m2 - 1) + (p3 - 1) * p3 ** (m3 - 1) + ...
```

Note that a ** b stands for the b'th power of a.

P38[39] (*) Compare the two methods of calculating Euler's totient function.

Use the solutions of problems P34 and P37 to compare the algorithms. Take the number of logical inferences as a measure for efficiency. Try to calculate phi(10090) as an example.

[35] http://picolisp.com/5000/!wiki?99p34
[36] http://picolisp.com/5000/!wiki?99p35
[37] http://picolisp.com/5000/!wiki?99p36
[38] http://picolisp.com/5000/!wiki?99p37
[39] http://picolisp.com/5000/!wiki?99p38

P39[40] (*) A list of prime numbers.

Given a range of integers by its lower and upper limit, construct a list of all prime numbers in that range.

P40[41] (**) Goldbach's conjecture.

Goldbach's conjecture says that every positive even number greater than 2 is the sum of two prime numbers. Example: 28 = 5 + 23. It is one of the most famous facts in number theory that has not been proved to be correct in the general case. It has been *numerically* confirmed up to very large numbers. Write a predicate to find the two prime numbers that sum up to a given even integer.

```
: (goldbach 28)
-> (5 23)
```

P41[42] (**) A list of Goldbach compositions.

Given a range of integers by its lower and upper limit, print a list of all even numbers and their Goldbach composition.

```
: (goldbach-list 9 20)
10 = 3 + 7
12 = 5 + 7
14 = 3 + 11
16 = 3 + 13
18 = 5 + 13
20 = 3 + 17
```

In most cases, if an even number is written as the sum of two prime numbers, one of them is very small. Very rarely, the primes are both bigger than say 50. Try to find out how many such cases there are in the range 2..3000.

Example (for a print limit of 50):

```
: (goldbach-list 1 2000 50)
992 = 73 + 919
1382 = 61 + 1321
1856 = 67 + 1789
1928 = 61 + 1867
```

[40] http://picolisp.com/5000/!wiki?99p39
[41] http://picolisp.com/5000/!wiki?99p40
[42] http://picolisp.com/5000/!wiki?99p41

P42[43] No task defined yet

P43[44] No task defined yet

P44[45] No task defined yet

P45[46] No task defined yet

Logic and Codes

P46[47] (**) Truth tables for logical expressions.

Define a function that takes a logical expression (a function of two variables) and prints the truth table.

```
: (truthTable '((A B) (and A (or A B))))
T T T
T NIL T
NIL T NIL
NIL NIL NIL
```

[43] http://picolisp.com/5000/!wiki?99p42
[44] http://picolisp.com/5000/!wiki?99p43
[45] http://picolisp.com/5000/!wiki?99p44
[46] http://picolisp.com/5000/!wiki?99p45
[47] http://picolisp.com/5000/!wiki?99p46

P51[48] No task defined yet

P52[49] No task defined yet

P53[50] No task defined yet

P74[51] No task defined yet

P75[52] No task defined yet

P76[53] No task defined yet

P77[54] No task defined yet

P78[55] No task defined yet

P79[56] No task defined yet

Miscellaneous Problems

P90[57] (**) Eight queens problem

This is a classical problem in computer science. The objective is to place eight queens on a chessboard so that no two queens are attacking each other; i.e., no two queens are in the same row, the same column, or on the same diagonal.

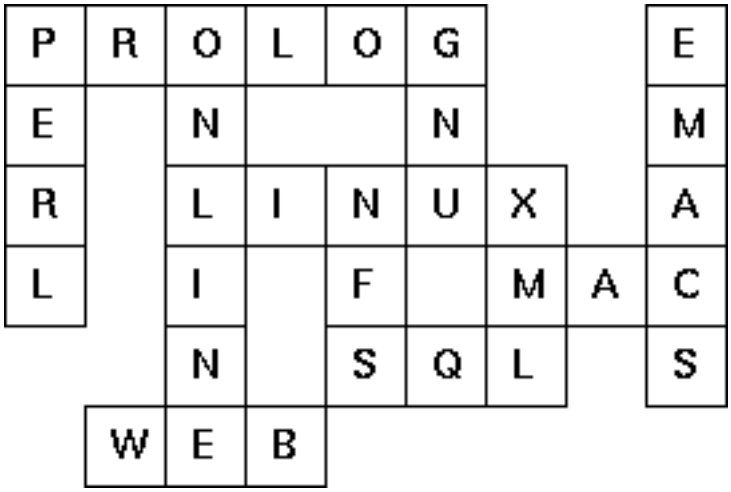
Hint: Represent the positions of the queens as a list of numbers 1..N.

Example: (4 2 7 3 6 8 5 1) means that the queen in the first column is in row 4, the queen in the second column is in row 2, etc. Use the generate-and-test paradigm.

P91[58] (**) Knight's tour

Another famous problem is this one: How can a knight jump on an NxN chessboard in such a way that it visits every square exactly once?

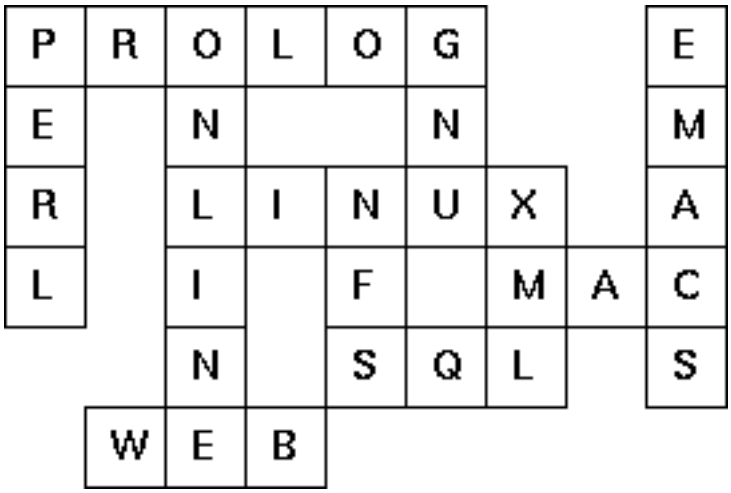
Hints: Represent the squares by pairs of their coordinates of the form X/Y, where both X and Y are integers between 1 and N. (Note that '/' is just a convenient functor, not division!) Define the relation jump(N,X/Y,U/V) to express the fact that a knight can jump from X/Y to U/V on a NxN chessboard. And finally, represent the solution of our problem as a list of N*N knight positions (the knight's tour).



P92[59] (***) Von Koch's conjecture

Several years ago I met a mathematician who was intrigued by a problem for which he didn't know a solution. His name was Von Koch, and I don't know whether the problem has been solved since.

Anyway the puzzle goes like this: Given a tree with N nodes (and hence N-1 edges). Find a way to enumerate the nodes from 1 to N and, accordingly, the edges from 1 to N-1 in such a way, that for each edge K the difference of its node numbers equals to K. The conjecture is that this is always possible.



For small trees the problem is easy to solve by hand. However, for larger trees, and 14 is already very large, it is extremely difficult to find a solution. And remember, we don't know for sure whether there is always a solution!

[48] http://picolisp.com/5000/!wiki?99p51
[49] http://picolisp.com/5000/!wiki?99p52
[50] http://picolisp.com/5000/!wiki?99p53
[51] http://picolisp.com/5000/!wiki?99p74
[52] http://picolisp.com/5000/!wiki?99p75
[53] http://picolisp.com/5000/!wiki?99p76
[54] http://picolisp.com/5000/!wiki?99p77
[55] http://picolisp.com/5000/!wiki?99p78
[56] http://picolisp.com/5000/!wiki?99p79
[57] http://picolisp.com/5000/!wiki?99p90
[58] http://picolisp.com/5000/!wiki?99p91
[59] http://picolisp.com/5000/!wiki?99p92

Write a predicate that calculates a numbering scheme for a given tree. What is the solution for the larger tree pictured above?

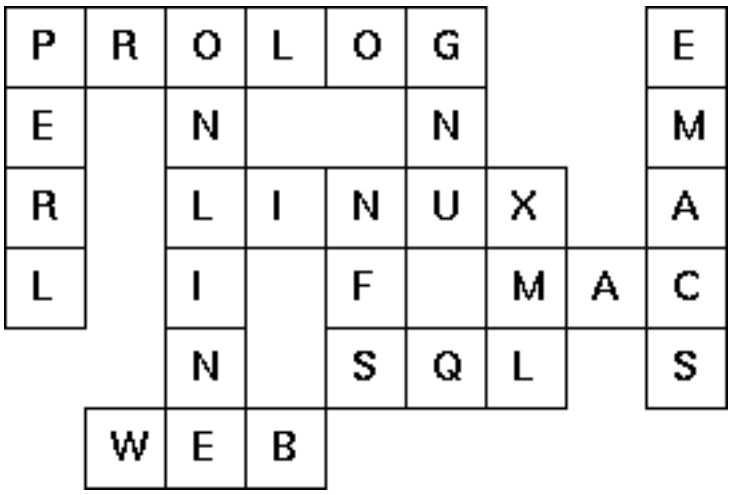
P93[60] (***) An arithmetic puzzle

Given a list of integer numbers, find a correct way of inserting arithmetic signs (operators) such that the result is a correct equation. Example: With the list of numbers (2 3 5 7 11) we can form the equations 2-3+5+7 = 11 or 2 = (3*5+7)/11 (and ten others!).

P95[61] (**) English number words

On financial documents, like cheques, numbers must sometimes be written in full words. Example: 175 must be written as "one hundred seventy-five". Write a function 'fullWords' to return (non-negative) integer numbers in full words.

P96[62] (**) Syntax checker



In a certain programming language (Ada) identifiers are defined by the syntax diagram (railroad chart) opposite. 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 function 'identifier' that can check whether or not a given string is a legal identifier.

P97[63] (**) 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
```

[60] http://picolisp.com/5000/!wiki?99pp93
[61] http://picolisp.com/5000/!wiki?99p95
[62] http://picolisp.com/5000/!wiki?99p96
[63] http://picolisp.com/5000/!wiki?99p97

```
        |         |                   |         |
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.

P98[64] (***) Nonograms

Around 1994, a certain kind of puzzles 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 PicoLisp 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.

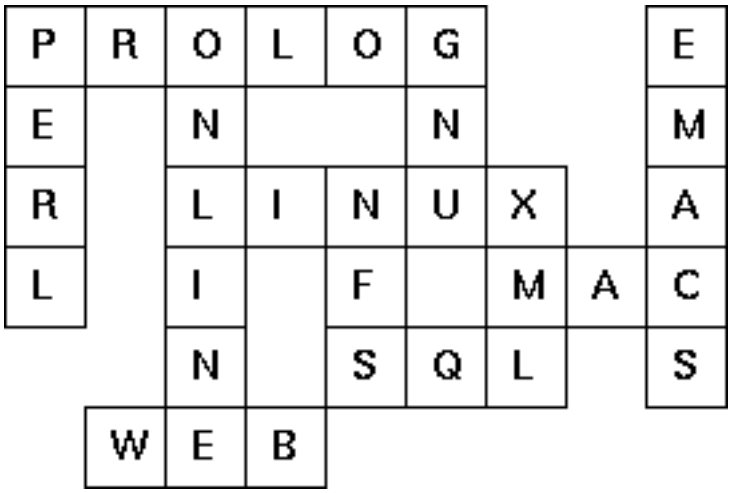
```
Problem statement:          Solution:

|_|_|_|_|_|_|_|_|_| 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.

[64] http://picolisp.com/5000/!wiki?99p98

P99[65] (***) 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 opposite is defined in the file p99a.dat[66], other examples are p99b.dat[67] and p99d.dat[68]. There is also an example of a puzzle (p99c.dat[69]) 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 (in Prolog?).

(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.

[65] http://picolisp.com/5000/!wiki?99p99
[66] http://picolisp.com/5000/!wiki?99p99a
[67] http://picolisp.com/5000/!wiki?99p99b
[68] http://picolisp.com/5000/!wiki?99p99d
[69] http://picolisp.com/5000/!wiki?99p99c