99 questions/21 to 28

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/) and Ninety-Nine Lisp Problems (http://www.ic.unicamp.br/~meidanis/courses/mc336/2006s2/funcional/L-99_Ninety-Nine_Lisp_Problems.html) .

1 Problem 21

Insert an element at a given position into a list.

Example:

```
* (insert-at 'alfa '(a b c d) 2)
(A ALFA B C D)
```

Example in Haskell:

```
P21> insertAt 'X' "abcd" 2
"aXbcd"
```

Solutions

2 Problem 22

Create a list containing all integers within a given range.

Example:

```
* (range 4 9)
(4 5 6 7 8 9)
```

Example in Haskell:

```
Prelude> range 4 9
[4,5,6,7,8,9]
```

Solutions

3 Problem 23

Extract a given number of randomly selected elements from a list.

Example:

```
* (rnd-select '(a b c d e f g h) 3)
(E D A)
```

Example in Haskell:

```
Prelude System.Random>rnd_select "abcdefgh" 3 >>= putStrLn
eda
```

Solutions

4 Problem 24

Lotto: Draw N different random numbers from the set 1..M.

Example:

```
* (rnd-select 6 49)
(23 1 17 33 21 37)
```

Example in Haskell:

```
Prelude System.Random>diff_select 6 49
Prelude System.Random>[23,1,17,33,21,37]
```

Solutions

5 Problem 25

Generate a random permutation of the elements of a list.

Example:

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

```
(B A D C E F)
```

Example in Haskell:

```
Prelude System.Random>rnd_permu "abcdef"
Prelude System.Random>"badcef"
```

Solutions

6 Problem 26

(**) 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.

Example:

```
* (combinations 3 '(a b c d e f))
((A B C) (A B D) (A B E) ... )
```

Example in Haskell:

```
> combinations 3 "abcdef"
["abc","abd","abe",...]
```

Solutions

7 Problem 27

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.

Example:

```
* (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.

Example:

```
* (group '(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".

Example in Haskell:

```
P27> group [2,3,4] ["aldo","beat","carla","david","evi","flip","gary","hugo","ida"]
[[["aldo","beat"],["carla","david","evi"],["flip","gary","hugo","ida"]],...]
(altogether 1260 solutions)

27> group [2,2,5] ["aldo","beat","carla","david","evi","flip","gary","hugo","ida"]
[[["aldo","beat"],["carla","david"],["evi","flip","gary","hugo","ida"]],...]
(altogether 756 solutions)
```

Solutions

8 Problem 28

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.

Example:

```
* (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))
```

Example in Haskell:

```
Prelude>lsort ["abc","de","fgh","de","ijkl","mn","o"]
Prelude>["o","de","de","mn","abc","fgh","ijkl"]
```

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.

Example:

```
* (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))
```

Example in Haskell:

```
lfsort ["abc", "de", "fgh", "de", "ijkl", "mn", "o"]
["ijkl","o","abc","fgh","de","de","mn"]
```

Solutions

Category:

- Tutorials
