# 99 questions/1 to 10

### From HaskellWiki

< 99 questions

This is part of Ninety-Nine Haskell Problems, based on Ninety-Nine Prolog Problems (https://sites.google.com/site/prologsite/prolog-problems) and Ninety-Nine Lisp Problems (http://www.ic.unicamp.br/~meidanis/courses/mc336/2006s2/funcional/L-99_Ninety-Nine_Lisp_Problems.html) .

## 1 Problem 1

(*) Find the last element of a list.

(Note that the Lisp transcription of this problem is incorrect.)

Example in Haskell:

```
Prelude> myLast [1,2,3,4]
4
Prelude> myLast ['x','y','z']
'z'
```

Solutions

## 2 Problem 2

(*) Find the last but one element of a list.

(Note that the Lisp transcription of this problem is incorrect.)

Example in Haskell:

```
Prelude> myButLast [1,2,3,4]
3
Prelude> myButLast ['a'..'z']
'y'
```

Solutions

# 3 Problem 3

(*) Find the K'th element of a list. The first element in the list is number 1.

Example:

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

Example in Haskell:

```
Prelude> elementAt [1,2,3] 2
2
Prelude> elementAt "haskell" 5
'e'
```

Solutions

# 4 Problem 4

(*) Find the number of elements of a list.

Example in Haskell:

```
Prelude> myLength [123, 456, 789]
3
Prelude> myLength "Hello, world!"
13
```

Solutions

# 5 Problem 5

(*) Reverse a list.

Example in Haskell:

```
Prelude> myReverse "A man, a plan, a canal, panama!"
"!amanap ,lanac a ,nalp a ,nam A"
Prelude> myReverse [1,2,3,4]
[4,3,2,1]
```

Solutions

# 6 Problem 6

(*) Find out whether a list is a palindrome. A palindrome can be read forward or backward; e.g. (x a m a x).

Example in Haskell:

```
*Main> isPalindrome [1,2,3]
False
*Main> isPalindrome "madamimadam"
True
*Main> isPalindrome [1,2,4,8,16,8,4,2,1]
True
```

Solutions

# 7 Problem 7

(**) 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).

Example:

```
* (my-flatten '(a (b (c d) e)))
(A B C D E)
```

Example in Haskell:

We have to define a new data type, because lists in Haskell are homogeneous.

```
 data NestedList a = Elem a | List [NestedList a]
*Main> flatten (Elem 5)
[5]
*Main> flatten (List [Elem 1, List [Elem 2, List [Elem 3, Elem 4], Elem 5]])
[1,2,3,4,5]
*Main> flatten (List [])
[]
```

Solutions

# 8 Problem 8

(**) 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.

Example:

```
* (compress '(a a a a b c c a a d e e e e))
(A B C A D E)
```

Example in Haskell:

```
> compress "aaaabccaadeeee"
"abcade"
```

Solutions

# 9 Problem 9

(**) Pack consecutive duplicates of list elements into sublists. If a list contains repeated elements they should be placed in separate sublists.

Example:

```
* (pack '(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))
```

Example in Haskell:

```
*Main> pack ['a', 'a', 'a', 'a', 'b', 'c', 'c', 'a',
             'a', 'd', 'e', 'e', 'e', 'e']
["aaaa","b","cc","aa","d","eeee"]
```

Solutions

# 10 Problem 10

(*) 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.

Example:

```
* (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))
```

Example in Haskell:

```
encode "aaaabccaadeeee"
[(4,'a'),(1,'b'),(2,'c'),(2,'a'),(1,'d'),(4,'e')]
```

Solutions


Category:

- Tutorials

---