99 questions/Solutions/4

From HaskellWiki

< 99 questions | Solutions

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

Contents

1 The simple, recursive solution

This is similar to the
length
from
Prelude
:

```
myLength           :: [a] -> Int
myLength []        =  0
myLength (_:xs)    =  1 + myLength xs
```

The prelude for haskell 2010 can be found here. (http://www.haskell.org/onlinereport/haskell2010/haskellch9.html#x16-1710009)

2 Same, but using an "accumulator"

```
myLength :: [a] -> Int
myLength list = myLength_acc list 0
        where
                myLength_acc [] n = n
                myLength_acc (_:xs) n = myLength_acc xs (n + 1)
```

3 Using foldl/foldr

```
myLength :: [a] -> Int
myLength1 =  foldl (\n _ -> n + 1) 0
```

```haskell
myLength2 =  foldr (\_ n -> n + 1) 0
myLength3 =  foldr (\_ -> (+1)) 0
myLength4 =  foldr ((+) . (const 1)) 0
myLength5 =  foldr (const (+1)) 0
myLength6 =  foldl (const . (+1)) 0
```

4 Zipping with an infinite list

We can also create an infinite list starting from 1. Then we "zip" the two lists together and take the last element (which is a pair) from the result:

```haskell
myLength :: [a] -> Int
myLength1 xs = snd $ last $ zip xs [1..] -- Just for fun
myLength2 = snd . last . (flip zip [1..]) -- Because point-free is also fun
myLength3 = fst . last . zip [1..] -- same, but easier
```

5 Mapping all elements to "1"

We can also change each element into our list into a "1" and then add them all together.

```haskell
myLength :: [a] -> Int
myLength = sum . map (\_->1)
```


Category:

- Programming exercise spoilers
