99 questions/Solutions/6

From HaskellWiki

< 99 questions | Solutions

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

```
isPalindrome :: (Eq a) => [a] -> Bool
isPalindrome xs = xs == (reverse xs)
isPalindrome' []  = True
isPalindrome' [_] = True
isPalindrome' xs  = (head xs) == (last xs) && (isPalindrome' $ init $ tail xs)
```

Here's one to show it done in a fold just for the fun of it. Do note that it is less efficient then the previous 2 though.

```
isPalindrome'' :: (Eq a) => [a] -> Bool
isPalindrome'' xs = foldl (\acc (a,b) -> if a == b then acc else False) True input
        where
        input = zip xs (reverse xs)
```

Another one just for fun:

```
isPalindrome''' :: (Eq a) => [a] -> Bool
isPalindrome''' = Control.Monad.liftM2 (==) id reverse
```

Or even:

```
isPalindrome'''' :: (Eq a) => [a] -> Bool
isPalindrome'''' = (==) Control.Applicative.<*> reverse
```

Here's one that does half as many compares:

```
palindrome :: (Eq a) => [a] -> Bool
palindrome xs = p [] xs xs
   where p rev (x:xs) (_:_:ys) = p (x:rev) xs ys
         p rev (x:xs) [_] = rev == xs
         p rev xs [] = rev == xs
```

Here's one using foldr and zipWith.

```
palindrome :: (Eq a) => [a] -> Bool
palindrome xs = foldr (&&) True $ zipWith (==) xs (reverse xs)
palindrome' xs = and $ zipWith (==) xs (reverse xs) -- same, but easier
```

```
isPalindrome list = take half_len list == reverse (drop (half_len + (len `mod` 2)) list)
        where
```

```
                len = length list
                half_len = len `div` 2

isPalindrome' list = f_part == reverse s_part
        where
                len = length list
                half_len = len `div` 2
                (f_part, s_part') = splitAt half_len list
                s_part = drop (len `mod` 2) s_part'
```

Using Control.Arrows (&&&) fan out operator.

With monomorphism restriction:

```
isPalindrome1 xs = (uncurry (==) . (id &&& reverse)) xs
```

Point free with no monomorphism restriction:

```
isPalindrome1 = (uncurry (==) . (id &&& reverse))
```

Retrieved from "https://wiki.haskell.org/index.php?title=99_questions/Solutions/6&oldid=59317"
Category:

- Programming exercise spoilers

- This page was last modified on 15 February 2015, at 13:29.
-