99 questions/Solutions/9

From HaskellWiki

< 99 questions | Solutions

(**) Pack consecutive duplicates of list elements into sublists.

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

```
pack (x:xs) = let (first,rest) = span (==x) xs
              in (x:first) : pack rest
pack [] = []
```

This is implemented as `group` in `Data.List`.

A more verbose solution is

```
pack :: Eq a => [a] -> [[a]]
pack [] = []
pack (x:xs) = (x:first) : pack rest
         where
           getReps [] = ([], [])
           getReps (y:ys)
                   | y == x = let (f,r) = getReps ys in (y:f, r)
                   | otherwise = ([], (y:ys))
           (first,rest) = getReps xs
```

Similarly, using **`splitAt`** and `findIndex`:

```
pack :: Eq a => [a] -> [[a]]
pack [] = []
pack (x:xs) = (x:reps) : (pack rest)
    where
        (reps, rest) = maybe (xs,[]) (\i -> splitAt i xs)
                             (findIndex (/=x) xs)
```

Another solution using **`takeWhile`** and **`dropWhile`**:

```
pack :: (Eq a) => [a] -> [[a]]
pack [] = []
pack (x:xs) = (x : takeWhile (==x) xs) : pack (dropWhile (==x) xs)
```

Or we can use
foldr
to implement this:

```
pack :: (Eq a) => [a] -> [[a]]
pack = foldr func []
    where func x []     = [[x]]
          func x (y:xs) =
              if x == (head y) then ((x:y):xs) else ([x]:y:xs)
```

A simple solution:

```
pack :: (Eq a) => [a] -> [[a]]
pack [] = []
pack [x] = [[x]]
pack (x:xs) = if x `elem` (head (pack xs))
              then (x:(head (pack xs))):(tail (pack xs))
              else [x]:(pack xs)

pack' [] = []
pack' [x] = [[x]]
pack' (x:xs)
    | x == head  h_p_xs = (x:h_p_xs):t_p_hs
    | otherwise         = [x]:p_xs
    where p_xs@(h_p_xs:t_p_hs) = pack' xs
myPack [] = []
myPack (y:ys) = impl ys [[y]]
        where
                impl [] packed = packed
                impl (x:xs) packed
                        | x == (head (last packed)) = impl xs ((init packed) ++ [x:(last pack
                        | otherwise                 = impl xs (packed ++ [[x]])

myPack' [] = []
myPack' (y:ys) = reverse $ impl ys [[y]]
        where
                impl [] packed = packed
                impl (x:xs) p@(z:zs)
                        | x == (head z) = impl xs ((x:z):zs)
                        | otherwise     = impl xs ([x]:p)
```

Retrieved from "https://wiki.haskell.org/index.php?title=99_questions/Solutions/9&oldid=59323"
Category:

- Programming exercise spoilers
