99 questions/Solutions/12

From HaskellWiki

< 99 questions | Solutions

(**) Decode a run-length encoded list.

Given a run-length code list generated as specified in problem 11. Construct its uncompressed version.

```
decodeModified :: [ListItem a] -> [a]
decodeModified = concatMap decodeHelper
    where
      decodeHelper (Single x)     = [x]
      decodeHelper (Multiple n x) = replicate n x
```

We only need to map single instances of an element to a list containing only one element and multiple ones to a list containing the specified number of elements and concatenate these lists.

A solution for the simpler encoding from problem 10 can be given as:

```
decode :: [(Int, a)] -> [a]
decode = concatMap (uncurry replicate)
```

This can be easily extended given a helper function:

```
toTuple :: ListItem a -> (Int, a)
toTuple (Single x)     = (1, x)
toTuple (Multiple n x) = (n, x)
```

as:

```
decodeModified :: [ListItem a] -> [a]
decodeModified = concatMap (uncurry replicate . toTuple)
```

a naïve solution with
foldl
:

```
decodeModified :: [ListItem a]-> [a]
decodeModified = foldl (\x y -> x ++ decodeHelper y) []
    where
        decodeHelper :: ListItem a -> [a]
        decodeHelper (Single x)     = [x]
        decodeHelper (Multiple n x) = replicate n x
```

foldl
can also be used to solve this problem:

```
decodeModified :: [ListItem a] -> [a]
decodeModified = foldl (\acc e -> case e of Single x -> acc ++ [x]; Multiple n x -> acc ++ re
```

Another way to decode the simplified encoding (which encoding, in the opinion of this editor, is a far more sensible one for Haskell):

```
decode :: Eq a => [(Int,a)] -> [a]
decode xs = foldr f [] xs
  where
    f (1, x) r = x : r
    f (k, x) r = x : f (k-1, x) r
```

Or, to make it a good transformer for list fusion,

```
{-# INLINE decode #-}
decode :: Eq a => [(Int,a)] -> [a]
decode xs = build (\c n ->
  let
    f (1, x) r = x `c` r
    f (k, x) r = x `c` f (k-1, x) r
  in
    foldr f n xs)
```


Category:

- Programming exercise spoilers
