99 questions/Solutions/18

From HaskellWiki

< 99 questions | Solutions

(**) Extract a slice from a list.

Given two indices, i and k, the slice is the list containing the elements between the i'th and k'th element of the original list (both limits included). Start counting the elements with 1.

```
slice xs i k | i>0 = take (k-i+1) $ drop (i-1) xs
```

The same solution as above, but the more paranoid (maybe too paranoid?) version of it (uses guards and Maybe):

```
slice :: [a] -> Int -> Int -> Maybe [a]
slice [] _ _ = Just []
slice xs k n    | k == n = Just []
                | k > n || k > length xs ||
                  n > length xs || k < 0 || n < 0 = Nothing
                | k == 0 = Just (take n xs)
                | otherwise = Just (drop (k-1) $ take n xs)
```

Or, an iterative solution:

```
slice :: [a] -> Int -> Int -> [a]
slice lst    1 m = slice' lst m []
        where
                slice' :: [a]->Int->[a]->[a]
                slice' _ 0 acc = reverse acc
                slice' (x:xs) n acc = slice' xs (n - 1) (x:acc)
                slice' [] _ _ = []
slice (x:xs) n m = slice xs (n - 1) (m - 1)
slice []     _ _ = []
```

Or:

```
slice :: [a] -> Int -> Int -> [a]
slice [] _ _  = []
slice (x:xs) i k
 | i > 1      = slice xs (i - 1) (k - 1)
 | k < 1      = []
 | otherwise  = x:slice xs (i - 1) (k - 1)
```

Another way using **splitAt**, though not nearly as elegant as the **take** and **drop**

version:

```
slice :: [a] -> Int -> Int -> [a]
slice xs i k = chunk
  where chop  = snd $ splitAt i' xs          -- Get the piece starting at i
        chunk = fst $ splitAt (k - i') chop  -- Remove the part after k
        i'    = i - 1
```

A little cleaner, using the previous problem's split (a.k.a.
splitAt
):

```
slice xs (i+1) k = snd (split (fst (split xs k)) i)
```

A solution using
zip
,
filter
then
map
seems straight-forward to me (*NB: this won't work for infinite lists*):

```
slice xs i j = map snd
               $ filter (\(x,_) -> x >= i && x <= j)
               $ zip [1..] xs
```

A solution using list comprehension:

```
slice xs i k = [x | (x,j) <- zip xs [1..k], i <= j]
```

Another simple solution using take and drop:

```
slice :: [a] -> Int -> Int -> [a]
slice l i k
  | i > k = []
  | otherwise = (take (k-i+1) (drop (i-1) l))
```

Zip, filter, unzip:

```
slice :: [a] -> Int -> Int -> [a]
slice xs a b = fst $ unzip $ filter ((>=a) . snd) $ zip xs [1..b]
```

Take and drop can be applied in the opposite order too:

```
slice xs i k = drop (i-1) $ take k xs
```

Using a fold:

```
slice :: [a] -> Int -> Int -> [a]
slice (x:xs) begin end = snd $ foldl helper (1, []) (x:xs)
    where helper (i, acc) x = if (i >= begin) && (i <= end) then (i+1, acc ++ [x]) else (i+1,
```

Retrieved from "https://wiki.haskell.org/index.php?title=99_questions/Solutions/18&oldid=58123"
Category:

- Programming exercise spoilers

- Recent content is available under a simple permissive license.