99 questions/Solutions/19

From HaskellWiki

< 99 questions | Solutions

(**) Rotate a list N places to the left.

Hint: Use the predefined functions length and (++).

```
rotate [] _ = []
rotate l 0 = l
rotate (x:xs) (n+1) = rotate (xs ++ [x]) n
rotate l n = rotate l (length l + n)
```

(Note that this solution uses n+k-patterns (http://en.wikibooks.org/wiki/Haskell/Pattern_matching#n.2Bk_patterns) which are removed (http://www.haskell.org/onlinereport/haskell2010/haskellli2.html#x3-5000) from Haskell 2010.)

There are two separate cases:

- If n > 0, move the first element to the end of the list n times.
- If n < 0, convert the problem to the equivalent problem for n > 0 by adding the list's length to n.

or using cycle:

```
rotate xs n = take len . drop (n `mod` len) . cycle $ xs
    where len = length xs
```

or without mod:

```
rotate xs n = take (length xs) $ drop (length xs + n) $ cycle xs
```

or

```
rotate xs n = if n >= 0 then
                  drop n xs ++ take n xs
              else let l = ((length xs) + n) in
                  drop l xs ++ take l xs
```

or

```
rotate xs n | n >= 0 = drop n xs ++ take n xs
            | n < 0 = drop len xs ++ take len xs
                      where len = n+length xs
rotate xs n = drop nn xs ++ take nn xs
    where
      nn = n `mod` length xs
```

Using a simple splitAt trick

```
rotate xs n
    | n < 0 = rotate xs (n+len)
    | n > len = rotate xs (n-len)
    | otherwise = let (f,s) = splitAt n xs in s ++ f
    where len = length xs
```

Without using
length
:

```
rotate xs n
    | n > 0 = (reverse . take n . reverse $ xs) ++ (reverse . drop n . reverse $ xs)
    | n <= 0 = (drop (negate n) xs) ++ (take (negate n) xs)
```

A much simpler solution without using
length
that is very similar to the first solution:

```
rotate :: [a] -> Int -> [a]
rotate [] _ = []
rotate x 0 = x
rotate x y
  | y > 0 = rotate (tail x ++ [head x]) (y-1)
  | otherwise = rotate (last x : init x) (y+1)
```


Category:

- Programming exercise spoilers
