99 questions/Solutions/20

From HaskellWiki

< 99 questions | Solutions

(*) Remove the K'th element from a list.

```
removeAt :: Int -> [a] -> (a, [a])
removeAt k xs = case back of
        [] -> error "removeAt: index too large"
        x:rest -> (x, front ++ rest)
  where (front, back) = splitAt (k - 1) xs
```

Simply use the
splitAt
to split after k - 1 elements.

If the original list has fewer than k elements, the second list will be empty, and there will be no element to extract. Note that we treat 1 as the first element in the list.

or

```
removeAt n xs = (xs !! (n - 1), take (n - 1) xs ++ drop n xs)
```

Another solution that avoids throwing an error and using ++ operators. Treats 1 as the first element in the list.

```
removeAt :: Int -> [a] -> (Maybe a, [a])
removeAt _ [] = (Nothing, [])
removeAt 1 (x:xs) = (Just x, xs)
removeAt k (x:xs) = let (a, r) = removeAt (k - 1) xs in (a, x:r)
```

Another solution that also uses Maybe to indicate failure:

```
removeAt :: Int -> [a] -> (Maybe a, [a])
removeAt _ [] = (Nothing, [])
removeAt 0 xs = (Nothing, xs)
removeAt nr xs  | nr > length xs = (Nothing, xs)
                | otherwise = (Just (xs !! nr), fst splitted ++ (tail . snd) splitted)
                      where splitted = splitAt nr xs
```

And yet another solution (without error checking):

```
removeAt :: Int -> [a] -> (a, [a])
removeAt n xs = let (front, back) = splitAt n xs in (last front, init front ++ back)
```

Similar, point-free style:

```
removeAt n = (\(a, b) -> (head b, a ++ tail b)) . splitAt (n - 1)
```

A simple recursive solution:

```
removeAt 1 (x:xs) = (x, xs)
removeAt n (x:xs) = (l, x:r)
        where (l, r) = removeAt (n - 1) xs
```


Category:

- Programming exercise spoilers
