99 questions/Solutions/3

From HaskellWiki

< 99 questions | Solutions

(*) Find the K'th element of a list. The first element in the list is number 1.

This is (almost) the infix operator !! in Prelude, which is defined as:

```
(!!)               :: [a] -> Int -> a
(x:_)  !! 0        =  x
(_:xs) !! n        =  xs !! (n-1)
```

Except this doesn't quite work, because !! is zero-indexed, and element-at should be one-indexed. So:

```
elementAt :: [a] -> Int -> a
elementAt list i    = list !! (i-1)
```

Or without using the infix operator:

```
elementAt' :: [a] -> Int -> a
elementAt' (x:_) 1  = x
elementAt' [] _     = error "Index out of bounds"
elementAt' (_:xs) k
  | k < 1           = error "Index out of bounds"
  | otherwise       = elementAt' xs (k - 1)
```

Alternative version:

```
elementAt'' :: [a] -> Int -> a
elementAt'' (x:_) 1  = x
elementAt'' (_:xs) i = elementAt'' xs (i - 1)
elementAt'' _ _      = error "Index out of bounds"
```

This does not work correctly on invalid indexes and infinite lists, e.g.:

```
elementAt'' [1..] 0
```

A few more solutions using prelude functions:

```
elementAt'' xs n | length xs < n = error "Index out of bounds"
           | otherwise = fst . last $ zip xs [1..n]
elementAt''' xs n = head $ foldr ($) xs
                         $ replicate (n - 1) tail
-- Negative indeces not handled correctly:
-- Main> elementAt''' "haskell" (-1)
-- 'h'
elementAt_w' xs n = last . take n $ xs -- wrong
-- Main> map (elementAt_w' [1..4]) [1..10]
-- [1,2,3,4,4,4,4,4,4,4]
```

```
elementAt_w'' xs n = head . reverse . take n $ xs -- wrong
-- Main> map (elementAt_w'' [1..4]) [1..10]
-- [1,2,3,4,4,4,4,4,4,4]
elementAt_w''' xs n = head . drop (n - 1) $ xs -- wrong
-- Main> map (elementAt_w''' [1..4]) [0..10]
-- [1,1,2,3,4,*** Exception: Prelude.head: empty list
```

or

`elementAt_w'`

correctly in point-free style:

```
elementAt_w'pf = (last .) . take . (+ 1)
```

Pedantic note: the above definition of

`elementAt_w'pf`

does not conform to the order of arguments specified by the question, but the following does:

```
elementAt_w'pf' = flip $ (last .) . take . (+ 1)
```


Category:

- Programming exercise spoilers
