Learn Physics with Functional Programming

—

Scott N. Walck

—

Chapter 5: Exercises

Graham Strickland

October 27, 2024

5.4 We have the function `range` with the following definition:

```
range :: Int -> [Int]
range x = if x >= 0
          then [0..x]
          else [x..0]
```

`range` returns a list containing all the integers between the argument (inclusive) and 0 in increasing order, i.e, range(2) = 0, 1, 2, range(−4) = −4, −3, . . . , 0, and range(0) = 0.

We demonstrate as follows:

```
ghci> range (-4)
[-4,-3,-2,-1,0]
ghci> range 2
[0,1,2]
ghci> range (-4)
[-4,-3,-2,-1,0]
ghci> range 0
[0]
```

5.5 We have the function `null'` with the following definition:

```
import Data.Foldable
```

```
null' :: (Foldable t) => t a -> Bool
null' xs = case toList xs of
  [] -> True
  (_ : _) -> False
```

null' returns True if an argument t of type a is empty, otherwise False.
Since we are using the Foldable type, we import Data.Foldable.

We demonstrate as follows:

```
ghci> null' []
True
ghci> null' [1, 2, 3]
False
ghci> null' [1..]
False
```

5.6 We have the funciton last' with the following definition

```
import GHC.Stack (HasCallStack)

last' :: HasCallStack => [a] -> a
last' x = head (reverse x)
```

last' returns the last item in an argument with type that implements
HasCallStack, an error if the argument is empty, or hangs indefinitely if
the variable has infinite length.

We demonstrate as follows:

```
ghci> last' [1, 2, 3]
3
ghci> last' ["check", "mate"]
"mate"
ghci> last' []
*** Exception: Prelude.head: empty list
CallStack (from HasCallStack):
  error, called at libraries/base/GHC/List.hs:1646:3
    in base:GHC.List
  errorEmptyList, called at libraries/base/GHC/List.hs:85:11
    in base:GHC.List
  badHead, called at libraries/base/GHC/List.hs:81:28
    in base:GHC.List
  head, called at last.hs:4:11 in main:Main
  last', called at <interactive>:4:1 in interactive:Ghci3
```