CS302: Paradigms of Programming

Lab 6: Programming in Haskell

June 1st, 2021

Q1. Try out the `commonWords` program from the class (check with some sample examples). Notice that we had started off by converting each letter in the text to lowercase and then separating out the words in the text. An alternative is to do the other way round: first separate out the words and then convert each letter to lowercase. The first method was expressed as `words . map toLower`. Give an expression for the second method and check that it works.

Q2. Check the types of existing Haskell functions `take` and `drop`, which are used to get first few and drop first few elements of a list, respectively. Define your own versions `myTake` and `myDrop` that work similar. Can you check if you have handled all corner cases (such as trying to take more elements than present in the list)?

Q3. Say we define an ADT in Haskell to construct lists by adding elements to the end (instead of to the front):

```
data List a = Nil | Snoc (List a) a
```

Note that `Snoc` here is like a backward `cons`. Thus, the list `[1,2,3]` in this view would be represented as `Snoc (Snoc (Snoc Nil 1) 2) 3`.

First define two functions `scar` and `scdr` that work like `car` and `cdr` for the snoc-view of lists, and then define two more functions for converting one view of lists to the other:

```
toList :: [a] -> List a
fromList :: List a -> [a]
```