15-150 Spring 2012
Lab 9

March 21, 2012

1 Introduction

In this lab, you will practice using sequences. Please take advantage of this opportunity to practice writing functions with the assistance of the TAs and your classmates. You are encouraged to collaborate with your classmates and to ask the TAs for help.

1.1 Getting Started

Update your clone of the `git` repository to get the files for this weeks lab as usual by running

```
git pull
```

from the top level directory (probably named `15150`).

1.2 Methodology

You must use the five step methodology for writing functions for every function you write on this assignment. In particular, every function you write should have a purpose and tests.

2 Survey

Task 2.1 Fill out the midsemester survey:

`www.surveymonkey.com/s/catpinski`

3 Sequences Cheat-Sheet

For your convenience a brief description of some of the functions on sequences is given here. See the Lecture 17 notes for more details.

- `Seq.map : ('a -> 'b) -> 'a Seq.seq -> 'b Seq.seq`, which takes a function and a sequence and returns a sequence whose elements are the result of applying the given function to the corresponding element in the given sequence.
- `Seq.reduce : (('a * 'a) -> 'a) -> 'a -> 'a Seq.seq -> 'a`, which combines all the elements of a sequence using a particular function and base case.
- `Seq.mapreduce : ('a -> 'b) -> 'b -> ('b * 'b -> 'b) -> 'a Seq.seq -> 'b`, which combines the operations of `Seq.map` and `Seq.reduce` by applying the given function of type `'a -> 'b` to each element of the sequence before combining them as in `Seq.reduce`.
- `Seq.filter : ('a -> bool) -> 'a Seq.seq -> 'a Seq.seq`, which computes the sequence that contains only those elements satisfying the given predicate.
- `Seq.length : 'a Seq.seq -> int`, which returns the number of elements in the sequence.
- `Seq.nth : int -> 'a Seq.seq -> 'a`, which returns the element of the given sequence at the indicated index, assuming it is in bounds.
- `Seq.tabulate : (int -> 'a) -> int -> 'a Seq.seq`, which computes a sequence of the given length such that the value of each element of the sequence is the result of applying the function to its index.
- `Seq.empty : unit -> 'a Seq.seq`, which forms an empty sequence.
- `Seq.cons : 'a -> 'a Seq.seq -> 'a Seq.seq`, which inserts the given element at the beginning of the sequence.
- `Seq.append : 'a Seq.seq -> 'a Seq.seq -> 'a Seq.seq`, which combines two sequences by inserting the elements of the second sequence after the elements of the first sequence.

4 Warm-Up

Recall the function `List.exists : ('a -> bool) -> 'a list -> bool`, which determines whether an element of the list satisfies the given predicate. You will write an analogous function for sequences:

Task 4.1 Write the function

```
seqExists : ('a -> bool) -> 'a Seq.seq -> bool
```

to determine if the sequence has an element that satisfies the given predicate. Hint: use `map`, `reduce`, and/or `mapreduce`.

5 Tabulate Puzzles

The following functions ask you to become familiar with `Seq.tabulate`, `Seq.length`, and `Seq.nth`.

5.1 Append

There is a function `Seq.append` that appends two sequences. Suppose there wasn't, and write

```
fun myAppend (s1 : 'a Seq.seq) (s2 : 'a Seq.seq) : 'a Seq.seq = ...
```

On sequences of length n and m, your solution should have $O(n + m)$ work and $O(1)$ span.

5.2 Transpose

Recall the function `transpose` from Homework 5:

```
transpose [[1,2,3],
           [4,5,6]]
  ==>
[[1,4],
 [2,5],
 [3,6]]
```

Write

```
fun transpose (s : 'a Seq.seq Seq.seq) : 'a Seq.seq Seq.seq = ...
```

that transposes a sequence of sequences. You may assume that s is rectangular, with dimensions $m \times n$, where $m, n > 0$. Your solution should have $O(m \times n)$ work and $O(1)$ span.

Have a TA check your code before proceeding!

6 Google Whack

A "Google Whack" is a two-word search phrase that returns exactly one result.[1] Past Google Whacks include "ambidextrous scallywags" and "disenthralled nimrod;" last semester, a student discovered "cylindrification Athenaioan".

In this problem, you will write a function `whack` to determine whether a pair of words is a Google Whack. We represent the WWW as a sequence of strings, where each string represents the text of a page; e.g.

```
val theWWW = seqFromList [
    "Ethers are a class of organic compounds that contain an ether group -- an",
    "hI thEres do you livEs at three maiN streEt?",
    "Doctor Who is a British science fiction television programme produced by the",
    ...]
```

6.1 Helpers

We have provided the following helpers:

```
(* Compute a sequence of all the words (separated by spaces)
   in the given page *)
val words : string -> string Seq.seq =
    seqFromList o (String.tokens (fn s => s = #" "))

(* Converts a bool to either 1 for true or 0 for false *)
fun boolToInt b = case b of true => 1 | false => 0

(* Determine if the given int is 1 *)
fun isOne n = case n of 1 => true | _ => false
```

6.2 BothHit

Task 6.1 Write the function

```
fun bothHit (word1 : string, word2 : string) : string Seq.seq -> bool =
```

Assuming `ws` is a sequence of words, `bothHit (word1,word2) ws` should return true iff both `word1` and `word2` occur in `ws`. You should use the `seqExists` function.

For example:

```
val true = bothHit ("hi", "there") (words "hi there how are you")
val false = bothHit ("hi", "there") (words "hi how are you")
```

[1] People sometimes also ask that each word return more than one hit individually.

6.3 SearchBoth

Task 6.2 Using `bothHit`, write the function

```
fun searchBoth (word1 : string, word2 : string) : string Seq.seq -> string Seq.seq = ...
```

Given a collection of Web pages `pages`, `searchBoth (word1,word2) pages)` should compute the sequence of all pages in which both words occur. You should use the `Seq.filter` function.

For example,

```
val 2 = Seq.length (searchBoth ("Doctor","Who") theWWW)
```

for `theWWW` defined in the SML file.

6.4 CountBoth

Task 6.3 Using `bothHit`, write the funtion

```
fun countBoth (word1 : string, word2 : string) : string Seq.seq -> int = ...
```

Given a collection of Web pages `pages`, `countBoth (word1,word2) pages` should compute the number of pages in which both words occur.

One solution would be `Seq.length o searchBoth(word1,word2)`. Try to find a solution that runs in 1 pass and does not generate any intermediate sequences.

6.5 Whack

Task 6.4 Using `countBoth`, write the function

```
fun whack (word1 : string, word2 : string) : string Seq.seq -> bool = ...
```

Given a collection of Web pages `pages`, `whack (word1,word2) pages` should return true if the words are a Google Whack, and return false otherwise.

Have a TA check your code before proceeding!

6.6 All Whacks

If you manage to finish all of that, here is a more free-form bonus problem:

Task 6.5 Write a function

```
fun allWhacks (www : string Seq.seq) : (string * string) Seq.seq = ...
```

that finds all of the Google Whacks on the `www`. Don't worry about efficiency; the point is to play with sequence functions. You may want to write a helper or two.