COMP105: Programming Paradigms
Lab Sheet 6

Filename: Lab6.hs

This lab covers higher order programming, with a focus on `fold`, `scan`, `takeWhile`, `dropWhile`, and `zipWith`.

1. **Folds and scans.**

   (a) Use `foldr` and the `*` operator to write a function `list_product` that multiplies all elements of a list together.

   (b) Use `foldr` and the `||` operator (or) to write a function `list_any` that takes a list of Bools, and returns True if any of the list elements are True.

   (c) Use `foldr` to write a function `product_of_evens` that takes a list of numbers, and multiplies all the even elements together.

   (d) Use `foldr` to write a function `lt10` that takes a list of numbers and returns the number of elements that are strictly less than 10.

   (e) Use `foldr` to write a function `smalls` that takes a string, and returns a string containing only the small (non-capital) letters of the input string.

   (f) Use `foldr` to write a function `sum_evens_odds` that takes a list of numbers, and returns a tuple `(a, b)` where `a` is the sum of the even elements in the list, and `b` is the sum of the odd elements of the list.

   (g) For each of the above functions, replace `foldr` with `scanr`. Check that you understand the output of the new function. Change the call back to `foldr` before you upload to Codegrade.

2. **takeWhile and dropWhile**.

   (a) Use `takeWhile` to write a function `leading_caps` that takes a string, and returns the elements before the first small letter of the string.

   (b) Use `dropWhile` to write a function `drop_caps` that takes a string, and returns all of the elements after (and including) the first small letter of the string.

   (c) Use `takeWhile` and `dropWhile` to write a function `split_on c string` that takes a character `c` and a string, and returns a pair `(before, after)`, where `before` contains everything before the first instance of `c`, and `after` contains everything after the first instance of `c`. The first instance of `c` in the string should be dropped if it exists.

(d) (*) A comma-separated string contains words separated by commas. Write a function `from_csv` that turns a comma-separated string into a list of words. So for example, `from_csv "foo,bar,baz"` should return `["foo", "bar", "baz"]`. The function `split_on` may be helpful here.

3. **zipWith.**

   (a) Use `zipWith` to write a function `mul_lists` that multiplies two lists together.

   (b) Use `zipWith` to write a function `and_lists` that takes two lists of Bools, and applies `&&` to each pair of boolean values.

   (c) Use `zipWith` to write a function `keep_or_zero` that takes a list `l1` of Ints and a list `l2` of Bools, and performs the following operation. For each element `x` in `l1`, if the corresponding element of `l2` is False, then the element in the output list should be zero, otherwise it should be `x`.

   (d) (*) Use `reverse`, `zipWith`, and `and` (returns True if a list of Bools only contains True) to write a function `is_palindrome` that returns true if a string is a palindrome.

Lab complete.