CS314 HW 3: Haskell

Spring 2019

1. Hopscotch

   Your first task is to write a function

   ```
   skips :: [a] -> [[a]]
   ```

   The output of skips is a list of lists. The first list in the output should be the same as the input list. The second list in the output should contain every second element from the input list, and the nth list in the output should contain every nth element from the input list.

   For example:

   ```
   skips "ABCD"       == ["ABCD", "BD", "C", "D"]
   skips "hello!"     == ["hello!", "el!", "l!", "l", "o", "!"]
   skips [1]          == [[1]]
   skips [True,False] == [[True,False], [False]]
   skips []           == []
   ```

   Note that the output should be the same length as the input.

2. Local maxima

   A local maximum of a list is an element of the list which is strictly greater than both the elements immediately before and after it. For example, in the list [2,3,4,1,5], the only local maximum is 4, since it is greater than the elements immediately before and after it (3 and 1). 5 is not a local maximum since there is no element that comes after it.

   Write a function `localMaxima :: [Integer] -> [Integer]` which finds all the local maxima in the input list and returns them in order.

   For example:

   ```
   localMaxima [2,9,5,6,1] == [9,6]
   localMaxima [2,3,4,1,5] == [4]
   localMaxima [1,2,3,4,5] == []
   ```

3. Histogram

   For this task, write a function `histogram :: [Integer] -> String` which takes as input a list of Integers between 0 and 9 (inclusive), and outputs a vertical histogram showing how many of each number were in the input list. You may assume that the input list does not contain any numbers less than zero or greater than 9 (that is, it does not matter what your function does if the input does contain such numbers). Your output must exactly match the output shown in the examples below.

```
histogram [1,1,1,5] ==

 *
 *
 *   *
==========
0123456789

histogram [1,4,5,4,6,6,3,4,2,4,9] ==

    *
    *
    * *
 ******  *
==========
0123456789
```

   Important note: If you type something like `histogram [3,5]` at the ghci prompt, you should see something like this: `"   * *\n==========\n0123456789\n"`

   This is a textual representation of the String output, including `\n` escape sequences to indicate newline characters. To actually visualize the histogram as in the examples above, use putStr, for example, `putStr (histogram [3,5])`.

Submission

Please submit a single file named `hw3.hs` on Sakai.