Programming languages (TC-2006)

Homework 11

In this homework, you will practice higher-order functions in the Haskell language. Please consider that the purpose of this homework is to allow you to practice and identify strengths and weaknesses. Then, implement these functions as requested and avoid using any built-in functions that already do what you are requested to implement. Since this homework focuses on higher-order functions, consider that in all the functions requested, you must use, at least, one-higher order function such as `map`, `foldl`, `filter`, generators, among others. Failing to use at least one higher-order function in each of the requested functions will cancel any points related to such a function.

1 `unique` (10%)

Write a function in Haskell that receives a String as input and returns a String that contains only unique characters (all repeated characters are removed).

2 `multiples` (10%)

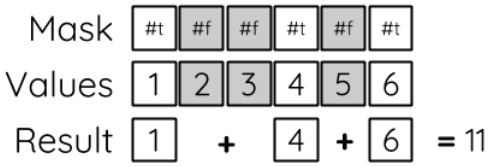
Write a function in Haskell that receives two parameters: a list of integers and an integer. The function returns a list with all the elements in the list which are multiples of the integer provided as second argument.

3 `add` (15%)

Write a function in Haskell that sums all the elements in a matrix of integers. For this assignment, use a 'by row' representation. For example, invoking the function with the argument `[[1, 2, 3], [4, 5, 6], [7, 8, 9]]` will return 45 (the sum of the numbers 1 to 9).

4 `maskedSum` (15%)

Write a function in Haskell that receives two lists of the same length, one of integers and the other one of booleans (there is no need for additional verifications). The function must sum all the elements that correspond to true values in the list of booleans. For example, invoking the function with the arguments `[1, 2, 3, 4, 5, 6]` and `[True, False, False, True, False, True]` will return 11 (the sum of the numbers 1, 4 and 6; the ones located at positions where the mask is `True`). A graphical representation of this example is depicted as follows:



5 `combine` (15%)

Implement a function in Haskell that receives two matrices as arguments (of a generic type) and returns a new matrix where each element contains the respective elements from the arguments (as a tuple).

For example, combining the following matrices[1]:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

Will produce the matrix:

$$\begin{pmatrix} (1,a) & (2,b) & (3,c) \\ (4,d) & (5,e) & (6,f) \\ (7,g) & (8,h) & (9,i) \end{pmatrix}$$

For representing the matrices, use a 'by row' representation.

6 `avgHeight` (15%)

In this problem, you will summarize data from a two-column table. The following table contains the heights of 10 randomly chosen students in a class, classified by gender:

Gender	Height (cms)
Male	178
Female	165
Female	158
Female	182
Male	161
Male	175
Female	179
Male	188
Male	169
Female	156

This table will be represented in Haskell as:

```
[
   ("Male", 178), ("Female", 165),
   ("Female", 158), ("Female", 182),
   ("Male", 161), ("Male", 175),
   ("Female",179), ("Male", 188),
   ("Male", 169), ("Female", 156)
]
```

Prepare a function in Haskell that returns the average of the heights per gender, and returns a tuple with such values properly labeled. For example, the result of calling the function on the previous table will be a pair of pairs: `(("Male",174.2),("Female",168.0))`.

[1] Please note that these matrices are not represented as they will be in Haskell.

7 `maxHeight` (20%)

Assume that now, the same table is represented by using a user-defined data type, as follows:

```
data Record = Record [Char] Double deriving Show
```

Then, the table will be represented in Haskell as:

```
[
   (Record "Male" 178), (Record "Female" 165),
   (Record "Female" 158), (Record "Female" 182),
   (Record "Male" 161), (Record "Male" 175),
   (Record "Female" 179), (Record "Male" 188),
   (Record "Male" 169), (Record "Female" 156)
]
```

Write a function that returns the maximum height per gender, and returns a tuple with such values properly labeled. For example, the result of calling the function on the previous table will be a pair of Records: `(Record "Male" 188.0,Record "Female" 182.0)`.

NOTE: For a maximum learning experience, try to solve this problem without using recursion or auxiliary functions.

Deliverables

Prepare an HS file that contains the functions requested and submit it to Canvas. **Please, do not submit other formats but RKT**. To prepare your HS file, use the code template distributed along with this document. The template contains some test cases for each function to help you verify that your codes work as requested.

I promise to apply my knowledge, strive for its development, and not use unauthorized or illegal means to complete this activity, following the Tecnológico de Monterrey Student Code of Honor.