Programming languages (TC-2006)

Extra points challenge 07

In this challenge, you will practice what you know from 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.

1 degree (20%)

Implement a function that calculates the degrees of all the nodes in a directed graph. Recall that the degree of a node is the number of nodes that can be reached directly from such a node. For this assignment, you are requested to use the following user-defined data type:

```haskell
data Graph = Graph [String] deriving Show
```

2 mode (20%)

Implement a function that calculates the mode on a list of integers. The mode is the value that appears most often in the data[1]. In case two or more values are the mode, the system will return the first one found. Please consider that the mode for an empty list is not defined and must produce an error. The result must be presented as a tuple, including both the value and its number of occurrences in the list.

3 Handling sets

For this exercise you will use lists to represent sets of integers. For example, `Set [1 2 4 5]` represents a set that contains the numbers 1, 2, 4 and 5. For this exercise you will need to implement three functions: `set`, `union`, and `intersection` for the following user-defined data type:

```haskell
data Set = Set [Int] deriving Show
```

3.1 set

(+1 ☆) This is the 'constructor' of the set. Since the sets do not allow repeated elements, and our representation is restricted to integers, this function takes a list on integers and removes repeated elements. For example, `Set [1, 1, 2, 3, 3, 1, 3]` must create a set with three elements: 1, 2, and 3. This set must comply with the user-defined data type described before.

3.2 union

(+1 ☆) This is the implementation of the union operator between two sets (by using the user-defined data type provided before). The union of sets A and B returns a set that contains all the elements contained in A and all the elements contained in B (of course, no repeated elements appear in the resulting set). Your function must guarantee that the resulting set is valid (contains no repeated elements).

[1] https://en.wikipedia.org/wiki/Mode_(statistics)

3.3 `intersection`

(+1 ✩) This is the implementation of the intersection operator between two sets (by using the user-defined data type provided before). The intersection of sets A and B returns a set that contains all the elements contained both in A and B. Your function must guarantee that the resulting set is valid (contains no repeated elements). You might find the function `elem` useful for implementing this function.

4 `mSort`

(+2 ✩) Write a program in Haskell that sorts a list of integers by using merge sort [2].

Deliverables

Prepare an HS file that contains the function requested and submit it to Canvas. **Please, do not submit other formats but HS.**

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.

[2] https://en.wikipedia.org/wiki/Merge_sort