Programming languages (TC-2006)

Homework 06

In this homework, you will practice with data structures implemented in the Racket 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 `sum` (10%)

Write a function in Racket that calculates the addition of all the elements within a matrix. For example, given the matrix:

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

the sum of all its elements would be 45.

2 `complete?` (10%)

By using the adjacency list representation for graphs, write a predicate in Racket that checks if a given graph is complete or not. Recall that a graph is said to be complete if every pair of distinct vertices in the graph is connected by a unique edge. For simplicity, assume that the graph provided as argument is in the proper format, so no further validations are required.

3 `msort` (15%)

Implement a function in Racket that sorts an array by using the mergesort algorithm [1]. You might find the functions `take` and `drop` useful for implementing this function[2].

4 `sold-units` (15%)

For some strange reason I do not fully understand (and at this point I am too afraid to ask), a sales manager used to keep all the information related to sales in a data structure implemented in Racket. The sales orders are represented as a table, where each record contains at least two elements (the sales order ID) and one or more lists of two elements (product ID and number of products sold). For example, the record:

```
'(100 (16 5) (10 3) (25 8))
```

indicates that five, three and eight units of products 16, 10 and 25 were sold in the sales order 100, respectively. Please note that, in the same order, there cannot be two entries related to the same product.

A new sales manager has just started working for the company but knows nothing of Racket or any other functional programming language. Your mission is to save the sales manager's day by implementing a function that counts the total number of units sold of a specific product, among all the orders. The product ID will be given as an argument to the function.

[1] `https://en.wikipedia.org/wiki/Merge_sort`

[2] Please take into consideration that these functions do not seem to work in some online Racket interpreters.

5 `insert` (20%)

By using the binary search tree representation discussed in class, propose a function in Racket that inserts an element into a binary search tree. Remember that inserting an element into a binary search tree must return a binary search tree (with the inserted element, of course).

6 Creating a data structure for handling sets

For this exercise you will use lists to represent sets of numbers. For example, the list `'(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 `intersaction`.

6.1 `set` (10%)

This is the 'constructor' of the set. Since the sets do not allow repeated elements, and our representation is restricted to numbers, this function takes a list and removes anything that is not a number and repeated elements. For example, `(set '(1 a '(3 5) 2 3))` must return `'(1 2 3)`.

6.2 `union` (10%)

This is the implementation of the union operator. 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 only numbers and no repeated elements).

6.3 `intersection` (10%)

This is the implementation of the intersection operator. 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 only numbers and no repeated elements).

Deliverables

Prepare an RKT file that contains the functions requested and submit it to Canvas. **Please, do not submit other formats but RKT**. To prepare your RKT 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.