Programming languages (TC-2006)

Extra points challenge 05

In this challenge, 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 `qsort` (25%)

Implement a function in Racket that sorts an array by using the quicksort algorithm [1].

2 `get-balance` (25%)

Imagine you want to calculate your monthly balance, given your incomes and outcomes. For some reason beyond understanding, you are tempted to use Racket to calculate this balance (unlikely, yet you will try it). Each operation will be registered as a list of three elements: type (string: "`income`", "`expense`"), description (string) and amount (numeric). All your operations will be contained in one list.

3 `toUpperTriangular` (25%)

A triangular matrix is a special kind of square matrix. A square matrix is called upper triangular if all the entries below the main diagonal are zero. Implement a function in Racket that, given a square matrix, returns its upper triangular matrix. For this assignment you are requested to use the matrix representation seen in class. For simplicity, assume that all the inputs are properly introduced so there are no additional verifications to be done.

For example, calling `upperTriangular` on the following 4×4 matrix:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix}$$

must return the matrix:

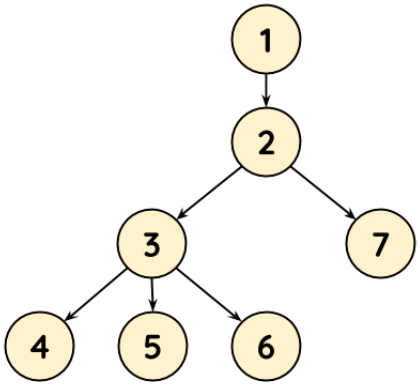
$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & 6 & 7 & 8 \\ 0 & 0 & 11 & 12 \\ 0 & 0 & 0 & 16 \end{pmatrix}$$

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

4 `dfs` (25%)

Write a function in Racket that receives an n-ary tree (by using one specific data structure) and traverses this tree by using depth-first search (DFS) [2]. Please note that you are not allowed to use `flatten` or any other built-in function that provides the requested behavior in order to traverse the tree.

For this problem, your function will receive an n-ary tree represented as lists of lists. A node in this tree contains a label and, all the remaining elements in the node represent its branches. Then, a leaf node will be represented as a list containing exactly one element (the label of the node). Please note that this representation differs significantly from the one seen in class. For example, the list `'(1 (2 (3 (4) (5) (6)) 7))` represents a tree as follows.



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.

[2] `https://en.wikipedia.org/wiki/Depth-first_search`