Programming languages (TC-2006)

Midterm Exam - Racket

Date: March 19, 2021

This exam contains four categories of problems. In the cases where a category contains more than one problem, you can choose which problem to solve. When you submit your solution, the selected problem per category must be implemented using the template distributed along with this document. Please be warned that failing to provide the output as requested will invalidate your solution. Also, do not remove the legend "Not yet implemented" unless you plan to solve the corresponding problem. In other words, it must be clear which problem you have chosen to solve. In case two problems are chosen in the same category, the points for that category will be canceled.

In all the cases, assume that the input is always in the proper format, so no additional validations are requested on the input. To be sure that your solution will get all the points, try the test cases contained in the template distributed along with this document. Please note that these test cases will be used for grading. Then, you are not requested to cover extreme cases, only the ones presented in the template. However, be warned that changing the names of the functions or their number of arguments, or failing to provide the output in the expected format described in the template will invalidate your solution.

Critical information: If your code does not run, your grade for the whole exam will be zero. **Double check that your code runs before submitting it to Canvas**.

1 Category I (25%)

1.1 `exam.takeif`

Write a function in Racket that, given a list, a predicate (f) and a number (n), retrieves the first n elements in the list that satisfy predicate f. Please note that you are not allowed to use the built-in `take` to implement this function.

1.2 `exam.dropif`

Write a function in Racket that, given a list, a predicate (f) and a number (n), removes the first n elements in the list that satisfy predicate f. Please note that you are not allowed to use the built-in `drop` to implement this function.

2 Category II (25%)

2.1 `exam.depth`

Provide a function in Racket that, given a binary search tree and a node, it indicates the depth of such a node. Please note that a root node has a depth of 0. For this problem, use the binary search tree representation seen in class, where a tree is represented as `(root (left) (rigth)`.

2.2 `exam.leaves`

Provide a function in Racket that, given a binary search tree and a node, it returns all the leaf nodes (the ones that have no child) as a list. For this problem, use the binary search tree representation seen in class, where a tree is represented as `(root (left) (right))`.

3 Category III (25%)

3.1 `exam.sdiag`

Write a function in Racket which receives a $n \times n$ square matrix M and returns the secondary diagonal as a list, i.e. all elements along the diagonal starting from the $M_{1,n}$ element, and going southwest, down to the $M_{n,1}$ element.

3.2 `exam.map`

Write a function in Racket that replicates the behavior of map (but limited to functions that take only one argument). Please note that, to implement this function, you are not allowed to use the built-in function `map`.

4 Category IV (25%)

Imagine you are given table with multiple columns. In Racket, the table will represented as nested lists, using one one list per record. The first row of such a table contains the information about the column names. For example, the following table contains information about a course:

id	g1	g2	g3
796623	62	84	91
782544	81	89	77
790256	59	62	63
799610	75	68	82
791313	85	56	92
786621	97	98	92
809706	91	72	80

Such a table would be coded in Racket as:

```
'(
   ("id" "g1" "g2" "g3")
   (796623 62 84 91)
   (782544 81 89 77)
   (790256 59 62 63)
   (799610 75 68 82)
   (791313 85 56 92)
   (786621 97 98 92)
   (809706 91 72 80)
)
```

4.1 `exam.select`

Write a function in Racket that receives a table (represented as in the previous example) and returns a table containing only the columns indicated in a list provided as argument.

4.2 `exam.insert`

Write a function in Racket that receives a table (represented as in the previous example) and inserts a column at a position specified by the user.

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.