Carleton University
Department of Systems and Computer Engineering
SYSC 3101 - Programming Languages - Winter 2023

Assignment 1

Posted: Sunday, Feb 5, 2023

Due: Sunday, Feb 19, 2023, 11:55 p.m.

References

Two documents at the Racket website provide plenty of information about the Racket dialect of Scheme:

The Racket Guide, https://docs.racket-lang.org/guide/index.html

The Racket Reference, https://docs.racket-lang.org/reference/index.html

A guide to the DrRacket IDE can be found here:

http://docs.racket-lang.org/drracket/index.html

Racket Coding Conventions

Please adhere to the conventions described in the Lab 1 handout.

"The Rules"

Do not use special forms that have not been presented in lectures. Specifically,

- Do not use `set!` to perform assignment; i.e., rebind a name to a new value.
- Do not use any of the Racket procedures that support *mutable* pairs and lists (`mpair`, `mcons`, `mcar`, `mcdr`, `set-mcar!`, `set-mcdr!`), as described in Section 4.10 of *The Racket Reference*.
- Do not use `begin` expressions to group expressions that are to be evaluated in sequence.

Submission

Submit your answers in only one Racket file(.rkt).

Stick to the names of the procedures in the questions, as parts of your assignment will be auto-graded using test cases.

Question 1:

(2.5 marks) Define a recursive procedure `count-multiples` that takes two arguments, a list of integers and an integer n, n ≥ 1. The procedure counts the number of elements in the list that are a multiple of n, by means of a recursive process. For example,

```
> (count-multiples '(1 2 3 4 5 6) 1) ; returns 6
> (count-multiples '(1 2 3 4 5 6) 2) ; returns 3
> (count-multiples '(1 2 3 4 5 6) 3) ; returns 2
> (count-multiples '(1 2 3 4 5 6) 7) ; returns 0
(define (count-multiples lst n)
```

Question 2:

(1.5 marks) Rewrite your solution to part (a) as procedure that generates an iterative process. Name this procedure `count-multiples-iter`

Question 3:

(6 marks) Define a procedure `deep-list-remove` that takes two inputs: a test procedure and a List.As output, it produces a List that is a copy of the input List with all of the elements for which the test procedure evaluates to true removed. Note the input list could be deep list.
For example,

```
> (deep-list-remove (lambda (x) (= x 0)) (list 0 1 2 3))
;returns (1 2 3).

> (deep-list-remove (lambda (x) (< x 4)) '(7 2 (3 4 (5 6))))
;returns '(7 ( 4 (5 6))).
```