CPSC 3740 – Spring 2021
Assignment 2
Due: March 3 (Wednesday), 11:55pm on Moodle

(0) Due to the COVID19 situation, there might be some adjustments to the assignment questions, if necessary.
(1) Please note that no collaboration is allowed for this assignment. Should any plagiarism be identified, all the involved students will get zero for the assignment and will be reported to the Chair of the department for further action.
(2) All the students are required to honor the academic integrity and exercise self-consciousness.

Part (B) (13 marks)

After you have installed DrRacket, launch DrRachet's user interface to start it.

Access the following website and see what it tells you about Dr. Racket and the programming language Scheme.

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

The tasks in this assignment are easy, just to familiarize yourself with Dr. Racket.

(1) Background introduction.
(2) Lexical and syntactical rules of the language.
(3) Various statements, such as conditionals, sequencings, iterations, etc.
(4) Various data types.
(5) Input and output

Help in DrRacket is always helpful, which takes you to the website of Dr. Racket. Frequently accessing it would be helpful.

Questions:

(1) (6 marks) Read the following web page about a function called *Ackermann*.

http://en.wikipedia.org/wiki/Ackermann_function

This function takes two parameters, *m* and *n*, and will calculate according to its definition.

Your implementation of the function should check whether the inputs are correct or not (for instance, *n* is a negative number or m is floating-point number). You need to output some error message if any input is not correct. Otherwise output *ackermann(m, n)*. (As indicated in the above webpage, do not try to test your function with "big" *m* or *n*.)

(2) (3 marks) A complete binary tree is defined as null, or a root or an internal node with exactly two children, each of which is a complete binary tree. In Scheme, we can represent such a tree using a list. One example is (*a* (*b* (*d*) (*e*)) (*c* (*f*) (*g*))), which means that the root of the tree is *a*, *b* and *c* are its left and right child, *d* and *e* are the left and right child of *b*, and *f* and *g* are the left and right child *c*, respectively. Assume that the input tree is always correct.

Read the following function called *x* and explain what it does.

```
(define x (lambda (atree)
    (if (pair? atree)
        (if (null? (cdr atree))
            1
            (+ (+ 1 (x (car (cdr atree)))) (x (car (cdr (cdr atree))))))
        0)))
```

(3) (4 marks) Continue from Question (2). Given such a complete binary tree represented as a list, create a function, called *preorder*, to traverse the tree using the pre order. The pre-order traversal of a binary tree is: *recursively,* starting from the root, access it, pre-order traverse the left child, and then pre-order traverse the right child. For the example, the result is a list (*a b d e c f g*).

Submission:

Your answers to (1), (2) and (3) should follow the format below.

; Answer to (1)
(define (ackermann …...)

; Answer to (2)
; Put your answer to Question 2 here

; Answer to (3)
(define (preorder ……)

The file to be saved for your answers should be called *YourLastName*.scm. Submit it on the Moodle.