CSCI 301, Winter 2020
Lab 3 (Racket Programming: Recursion and Set Operations)
DUE: 11:59pm Wednesday, 2/5, Online submission
25 Points Total

- This is an **individual** assignment. Work through the following lab.
- In this lab assignment, you will work experimentally with the DrRacket language on **Recursions and Set Operations**.
- Keep in mind that in addition to this lab, there are Racket and Scheme resources linked in the syllabus if you need help.

In Racket sets can be represented as lists. However, unlike lists, the order of values in a set is not significant. Thus both (1 2 3) and (3 2 1) represent the same set.

***For the following questions, you may assume that a sets contains only atomic values (numbers, string, symbols, etc. but not sets or lists) and it does not contain duplicate members.

1. Write a Racket function `(member? x L)` that tests whether $x \in L$ where L is a set (represented as a list). (Hint: $x \in L$ if and only if either `x` is equal to the head of `L`, or `x` is in the remainder of `L`.)
   Test cases:

```
(member? 1 '(3 2 1)) ---> #t
(member? 4 '(3 2 1)) ---> #f
(member? 1 '())---> #f
(member? 'susan '(susan john ryan)) ---> #t
```

2. Write a Racket function `(subset? L1 L2)` that tests whether $L_1 \subseteq L_2$. L1 is a subset of L2 if every element of L1 is also a member of L2.
   Test cases:

```
(subset? '(1 2 3) '(3 2 1))---> #t
(subset? '(1 2 3) '(4 5 6)) ---> #f
(subset? '(1 2 3) '(1 2 3 4 5 6)) ---> #t
(subset? '(1 2) '())---> #f
```

   ***Use the function `(member? x L)` as a helper function in your implementation.

3. Write a Racket function `(set-equal? L1 L2)` that tests whether L1 and L2 are equal. Two sets are equal if they contain exactly the same members, ignoring ordering (or in other words, two sets are equal if they are a subset of each other). For example

```
(set-equal? '(1 2 3) '(3 2 1)) ---> #t
(set-equal? '(1 2) '(3 2 1)) ---> #f
(set-equal? '(ryan susan john) '(susan john ryan)) ---> #t
```

4. Two common operations on sets are **union** and **intersection**. The union of two sets is the set of all elements that appear in either set (with no repetitions). The intersection of two sets is the set of elements that appear in both sets.
   Write Racket functions `(union S1 S2)` and `(intersect S1 S2)` that implement set union and set intersection.
   Test cases:

```
(union '(1 2 3) '(3 2 1)) ---> (1 2 3)
(union '(1 2 3) '(3 4 5)) ---> (1 2 3 4 5)
(union '(a b c) '(3 2 1)) ---> (a b c 1 2 3)
(intersect '(1 2 3) '(3 2 1)) ---> (1 2 3)
(intersect '(1 2 3) '(4 5 6)) ---> ()
(intersect '(1 2 3) '(2 3 4 5 6)) ---> (2 3)
```

   The ordering of the elements in your answer may differ from the above.

You must use recursion, and not iteration. You may not use side-effects (e.g. set!).

The solutions will be turned in by posting a single Racket program (lab03. rkt) containing a definition of all the functions specified.