CSCI 301, Summer 2019
Lab 5 (Properties of Relations)
DUE: 11:59pm Monday, 8/5, Online submission
30 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 **Properties of relations**.
- Keep in mind that in addition to this lab, there are Racket and Scheme resources linked in the syllabus if you need help.

Implement the following Racket functions:

1. **Reflexive-Closure**
   Input: a list of pairs, **L** and a list **S**. Interpreting **L** as a binary relation over the set **S**, **Reflexive-Closure** should return the reflexive closure of L.
   Examples:

```
(Reflexive-Closure '((a a) (b b) (c c)) '(a b c))
    ---> '((a a) (b b) (c c))
(Reflexive-Closure '((a a) (b b)) '(a b c))
   ---> '((a a) (b b) (c c))
(Reflexive-Closure '((a a) (a b) (b b) (b c)) '(a b c))
   ---> ((a a) (a b) (b b) (b c) (c c))
(Reflexive? '() '(a b c))
   ---> '((a a) (b b) (c c))
```

2. **Symmetric-Closure**
   Input: a list of pairs, **L**. Interpreting **L** as a binary relation, **Symmetric-Closure** should return the symmetric closure of L.

   Examples:

```
(Symmetric-Closure '((a a) (a b) (b a) (b c) (c b)))
    ---> '((a a) (a b) (b a) (b c) (c b))
(Symmetric-Closure '((a a) (a b) (a c)))
    ---> '((a a) (a b) (a c) (b a) (c a))
(Symmetric-Closure '((a a) (b b)))
    ---> '((a a) (b b))
(Symmetric-Closure '())
    ---> '()
```

3. **Transitive-Closure**
   Input: a list of pairs, **L**. Interpreting **L** as a binary relation, **Transitive-Closure** should return the transitive closure of L.
   Examples:

```
(Transitive-Closure '((a b) (b c) (a c)))
   ---> '((a b) (b c) (a c))
(Transitive-Closure '((a a) (b b) (c c)))
   ---> '((a a) (b b) (c c))
(Transitive-Closure '((a b) (b a)))
   ---> '((a b) (b a) (a a) (b b)))
(Transitive-Closure '((a b) (b a) (a a)))
   ---> '((a b) (b a) (a a) (b b))
(Transitive-Closure '((a b) (b a) (a a) (b b)))
   ---> '((a b) (b a) (a a) (b b))
(Transitive-Closure '())
   ---> '()
```

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 (lab05. rkt) containing a definition of all the functions specified.