**Lab 5**

**Properties of Relations**

This is an individual assignment. In this lab assignment, you will work with the DrRacket language on Properties of relations.

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

\*\*\*You can assume that the input lists don’t contain duplications

**Part I** (5 points per question)

Implement the following Racket functions:

1. **Reflexive?**

Input: a list of pairs, **L** and a list **S**. Interpreting **L** as a binary relation over the set **S**,  **Reflexive?** Returns **#t** if **L** is a reflexive relation over the set **S** and **#f** otherwise. Examples:

(display “Reflexive?\n”)

(Reflexive? ‘((a a) (b b) (c c)) ‘(a b c)) -à #t

(Reflexive? ‘((a a) (b b)) ‘(a b c)) -à #f

(Reflexive? ‘((a a) (a s) (b b) (c c)) ‘(a b c)) -à #f (Reflexive? ‘() ‘()) -à #t

**2. Symmetric?**

Input: a list of pairs, **L**. Interpreting **L** as a binary relation, **Symmetric?** Returns **#t** if **L** is a  symmetric relation and **#f** otherwise.

Examples:

(display “Symmetric?\n”)

(Symmetric? ‘((a a) (a b) (b a) (b c) (c b))) -à #t

(Symmetric? ‘((a a) (a b) (a c) (c a))) -à #f

(Symmetric? ‘((a a) (b b))) -à #t

(Symmetric? ‘()) -à #t

**3. Transitive?**

Input: a list of pairs, **L**. Interpreting **L** as a binary relation, **Transitive?** Returns **#t** if **L** is a  transitive relation and **#f** otherwise.

Examples:

(display “Transitive? \n”)

(Transitive? ‘((a b) (b c) (a c))) -à #t

(Transitive? ‘((a a) (b b) (c c))) -à #t

(Transitive? ‘((a b) (b a))) -à #f

(Transitive? ‘((a b) (b a) (a a))) -à #f

(Transitive? ‘((a b) (b a) (a a) (b b))) -à #t

(Transitive? ‘())-à #t

**Submission**

Prepare a single Racket program file (lab5.rkt) containing definitions of all the requested functions. Please make sure to use the requested function names and use comments to explain the parts that are hard to understand. Submit the file on Canvas.