**Lab 6**

**Closure of Relations**

This is an individual assignment. In this lab assignment, you will work with the DrRacket language on a set of relation properties.

\*\*\* 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** (6 points per question)

Implement the following Racket functions:

1. **Reflexive-Closure**

**Definition**:

The reflexive closure of a binary relation R on a set A is obtained by adding the elements (a, a) to the original relation R for all a ∈ A.

**Example**:

Consider the relation R = {(1, 2), (2, 4), (3,3), (4, 2)} on the set A = {1, 2, 3, 4}. R is not reflexive. To get the reflexive closure, we need to union R with a set containing all missing diagonal elements:

I = {(1, 1), (2, 2), (3, 3), (4, 4)}

R U I = {(1, 2), (2, 4), (3,3), (4, 2), (1, 1), (2, 2), (3, 3), (4, 4)}

**Input**:

A list of pairs **L** and a list **S**. Interpreting **L** as the 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**

**Definition:**

Given a relation R on a set A, the symmetric closure of R, denoted R', can be found as follows:

1. Start with the original relation R.
2. For each pair (a, b) ∈ R, ensure that the pair (b, a) is also included in the relation.
3. Combine all such pairs to form the symmetric closure.

If R is a relation on a set A, then the symmetric closure of R is RU {(b, a) | (a, b) ∈ R}

**Example**:

Consider the relation R = {(1, 2), (2, 4), (3,3), (4, 2)} on the set A = {1, 2, 3, 4}. The symmetric closure is as follows:

{(1, 2), (2, 4), (3,3), (4, 2), (2, 1), (4, 2)}

**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**

**Definition:**

To find the transitive closure of a given relation R, you need to:

* + 1. start with the original relation R.
    2. add pairs to R to make it transitive. Specifically, for any pairs (a, b) and (b, c) in R, add the pair (a, c) if itis not already in R.

**Example**:

Consider the relation R = {(1, 2), (2, 4), (3,3), (4, 2)} on the set A = {1, 2, 3, 4}. The symmetric closure is as follows:

{(1, 2), (2, 4), (3,3), (4, 2), (1, 4), (2, 2), (4, 4)}

Note:

* Because of (4, 2) and (2, 4), you need to include (4, 4) in the closure.
* Because of (2, 4) and (4, 2), you need to include (2, 2) in the 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 '())

 ---> '()

**Submission**

Prepare a single Racket program file (lab6.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.