Homework 15: Due Wednesday, May 6

Problem 1: Let A be a set and let R be a binary relation on A. The *symmetric closure* of R is the smallest binary relation S on A such that $R \subseteq S$ and S is symmetric. For example if $A = \{1, 2, 3, 4\}$ and

$$R = \{(1, 1), (2, 1), (2, 3), (3, 2), (4, 2), (4, 4)\}$$

then the symmetric closure of R is

$$S = \{(1, 1), (1, 2), (2, 1), (2, 3), (3, 2), (4, 2), (2, 4), (4, 4)\}$$

Write a Scheme program that takes in a relation R, and outputs the symmetric closure of R. Notice that this program can be used to form a the natural graph associated with a directed graph (i.e. by throwing in all "reverse" arrows that don't appear in the previous relation).

Problem 2: Let D be a directed graph with vertex set V.
a. In class, we talked about defining a new relation on V by letting $a \sim b$ mean that there exists an a, b-walk in D. Write a Scheme program that takes as input a directed graph D, and outputs this relation.
b. Now define a different relation on V by letting $a \sim b$ mean that there exists an a, b-walk in D of length at least 1. In other words, we do not automatically have that this relation is reflexive by definition, but we would still have $a \sim a$ if there is a cycle involving a. Write a Scheme program that takes as input a directed graph D, and outputs this relation. Briefly explain the conceptual reason why your program works.
Note: One can use part b to compute the so-called *transitive closure* of a relation, which is the smallest relation containing a given one that is transitive.

Problem 3: Write a Scheme program that takes as input a directed graph D, and outputs a Boolean according to whether there exists a vertex a that can reach every vertex of D. More formally, if D has vertex set V, then your program should output #t if there exists $a \in V$ such that

$$\{b \in V : \text{There exists an } a, b\text{-walk in } D\} = V,$$

and should output #f otherwise.

Problem 4: In class, we talked about a different way to code graphs other than through relations. The idea is that we will code them as a list of lists, where the first element of each list gives the vertices, and the remaining element of each list gives the vertices that appear as the endpoints of arrows starting with the initial vertex. For example, given our directed graph

```
(cons (interval-set 1 8) (list (cons 1 2) (cons 2 3) (cons 3 4) (cons 4 5)
                               (cons 3 8) (cons 1 7) (cons 7 5) (cons 6 7))))
```

from class, we could now code it as

$$'((1\ 2\ 7)\ (2\ 3)\ (3\ 4\ 8)\ (4\ 5)\ (5)\ (6\ 7)\ (7\ 5)\ (8)).$$

Write a Scheme program that takes a directed graph represented in the first form, and outputs a representation in this latter form.