

CS 101 Computer Science I (Spring 2001)
Lab 9: April 18-20, 2001

1. Write a procedure called `"set-cross-op"` that takes a procedure `fn` as its argument and returns a procedure `(set-cross-op fn)` as its result. The procedure `fn` takes two arguments. The procedure `(set-cross-op fn)` takes two sets `s1` and `s2` as arguments. The expression `((set-cross-op fn) s1 s2)` evaluates to a set consisting of the results of applying `fn` to all possible combinations of an element of `s1` and an element of `s2`. Use the `cartesian-product` procedure in your solution. (You may also use any of the other procedures in the file `"cs101-sets.scm"`.)

```
((set-cross-op +) (make-set 0 1) (make-set 10 15 20))
  → ("set" 10 15 20 11 16 21)
((set-cross-op *) (make-set 1 2) (make-set 10 15 20))
  → ("set" 10 15 20 20 30 40)
((set-cross-op list) (make-set 1 2) (make-set 10 20))
  → ("set" (1 10)(1 20)(2 10)(2 20))
```

2. Write a definition of a procedure called `"generate-relation"`. This procedure takes two sets called `"s1"` and `"s2"` as parameters. It also takes a binary predicate called `"r?"` as a parameter. The procedure `generate-relation` returns the set of all ordered pairs `(make-op x y)` such that `x` is a member of `s1`; `y` is a member of `s2`; and `(r? x y)` evaluates to `#t`. (You may use any of the procedures in the file `"cs101-sets.scm"`.)

```
(define s1 (make-set 1 2 3 4))
(define s2 (make-set 1 2 3 4))
(generate-relation s1 s2 <)
  → ("set" (1 2) (1 3) (1 4) (2 3) (2 4) (3 4))
```

3. A relation `R` is said to be *reflexive* on a set `S` if and only if for each element `e` of set `S`, the ordered pair `(e,e)` is in relation `R`. Write a definition of a Scheme procedure called `"reflexive?"`. This procedure takes a relation `r` and a set `s` as its arguments. It returns `#t` if `r` is reflexive on `s`, and returns `#f` otherwise. (You may use any of the procedures in the file `"cs101-sets.scm"`.)

```
(reflexive? (make-set (make-op 1 1)(make-op 1 2)(make-op 2 2))
            (make-set 1 2))    → #t
(reflexive? (make-set (make-op 1 1)(make-op 1 2))
            (make-set 1 2))    → #f
(reflexive? (make-set (make-op 1 1)(make-op 2 3)(make-op 2 2))
            (make-set 1 2 3))  → #f
```

4. A relation `R` is said to be symmetric if and only if for each ordered pair `(x,y)` in relation `R`, the ordered pair `(y,x)` is also in relation `R`. Write a definition of a Scheme procedure called `"symmetric?"`. This procedure takes a relation `r` as its argument. It returns `#t` if `r` is symmetric and returns `#f` otherwise. (You may use any of the procedures in the file `"cs101-sets.scm"`.)

```
(symmetric? (make-set (make-op 1 2) (make-op 2 1)
                     (make-make-op 2 3) (make-op 3 2))) → #t
(symmetric? (make-set (make-op 1 2) (make-op 2 3))) → #f
```