Task 1: Transposing Binary Relations [Scheme, 8 points]

A binary relation R over sets X and Y is a subset $R \subseteq X \times Y$, i.e., any set of ordered pairs (x, y) where $x \in X$ and $y \in Y$. Transpose R^T of R is a binary relation over Y and X defined as follows:

$$R^T = \{(y, x) \mid (x, y) \in R\}.$$

In this task, you are supposed to transpose a given relation. To represent a binary relation $R \subseteq X \times Y$ in Scheme, we introduce the following structure:

```
(struct entry (key vals) #:transparent)
```

The first component `key` represents an element $x \in X$ and the second component `vals` is the list of all $y \in Y$ such that $(x, y) \in R$. So we can store any relation $R \subseteq X \times Y$ as a list of entries.

Example

Let $X = \{1, 2, 3\}$, $Y = \{a, b, c, d, e\}$, and

$$R = \{(1, a), (1, b), (1, c), (1, d), (2, b), (2, d), (2, e), (3, c), (3, d)\}.$$

The relation R is represented as follows:

```
(define rel (list (entry 1 '(a b c d))
                  (entry 2 '(b d e))
                  (entry 3 '(c d))))
```

Implement a function `(transpose rel)` which takes a binary relation `rel` and computes its transpose. The resulting list of entries must be sorted by `key` and the list of values `vals` inside entries must be sorted as well. You may assume that the given relation `rel` relates integers and symbols as in the above example.

Examples

```
> (transpose (list (entry 1 '(a b c))))
(list (entry 'a '(1)) (entry 'b '(1)) (entry 'c '(1)))

> (transpose (list (entry 1 '(a)) (entry 2 '(a))))
(list (entry 'a '(1 2)))

> (transpose rel)
(list (entry 'a '(1))
      (entry 'b '(1 2))
      (entry 'c '(1 3))
      (entry 'd '(1 2 3))
      (entry 'e '(2)))
```

Your file has to be called `task1.rkt` and must provide the function `transpose` and the structure `entry`. It should start like this:

```
#lang racket
(provide transpose (struct-out entry))
(struct entry (key vals) #:transparent)

; your code goes here
```

Hint

To sort a list, use the function `(sort lst cmp)` whose second argument `cmp` is a function comparing two elements of `lst`, e.g.,

```
> (sort '(4 2 3 1) <)
'(1 2 3 4)

> (sort '(d b c a) symbol<?)
'(a b c d)
```

To extract the keys of the transpose relation, you might find it helpful to use functions `flatten` and `remove-duplicates`.