Task 1: Robot Karel [Scheme, 8 points]

Implement a function `(eval prg st)` evaluating a simple program `prg` for a robot called Karel in a state `st`. The function evaluates a sequence of commands for the robot and returns its final state after applying all the commands. The state of the robot is captured by the following structure:

```
(struct state (x y dir) #:transparent)
```

where `x` and `y` determine the position of the robot and `dir` is a direction, i.e., one of the symbols `'north 'west 'south 'east`.
The robot can execute three possible commands: `'left 'right 'move`. The command `'left` changes the robot's direction in the counter-clockwise direction, e.g. `'north` is changed to `'west`. Analogously, the command `'right` changes the robot's direction in the clockwise direction, e.g. `'north` is changed to `'east`. Finally, the command `'move` moves the robot by one step in its direction. More precisely, executing `'move` results in the following transitions depending on the robot's direction:

```
(state x y 'north) -> (state x (+ y 1) 'north)
(state x y 'south) -> (state x (- y 1) 'south)
(state x y 'east) -> (state (+ x 1) y 'east)
(state x y 'west) -> (state (- x 1) y 'west)
```

Implementation

Make sure your file is called `task1.rkt` and starts with:

```
#lang racket
(provide eval state)
(struct state (x y dir) #:transparent)
```

Example 1:

```
(eval '(move move left) (state 0 0 'north))
(state 0 2 'west)
```

Example 2:

```
(eval '(right move move move left move move left move) (state 3 5 'south))
(state 1 3 'east)
```