Tasks 3 and 4: Multiplayer ***Rock Paper Scissors***

Determine the winner (or remaining players) after several rounds of multiplayer Rock Paper Scissors.

Rules of multiplayer RPS

Players stand in a circle and all throw at once. If rock, paper, and scissors are all thrown, it is a stalemate, and they rethrow. If only two throws are present, all players with the losing throw are eliminated. The following function decides which throws from a set of set of ***throws*** in a single round should be eliminated.

$$\text{eliminated}(\text{throws}) \equiv \begin{cases} \{\texttt{rock}\} & \{\texttt{paper}, \texttt{rock}\} = \text{throws} \\ \{\texttt{paper}\} & \{\texttt{paper}, \texttt{scissors}\} = \text{throws} \\ \{\texttt{scissors}\} & \{\texttt{rock}, \texttt{scissors}\} = \text{throws} \\ \emptyset & \text{otherwise} \end{cases}$$

Data

The actions of players are decided in advance; e.g. for two rounds of three-player RPS, the players and actions are represented as:

```
(define players '("alice" "bob" "charlie"))
(define strategies '((r p) (r r) (s p)))
```

where Alice throws Rock in the first round, then Paper in the second. Charlie never gets to throw his second pick (Paper), as he will be eliminated after one round. Alice is the only remaining player after the two rounds. The result should be `'("alice")`.

Hints!

You will need to keep track of the players and their actions. In each round, figure out which throws should be eliminated, then filter out the corresponding players and their strategies. Create two helper functions: one that creates a boolean "mask" of all the winners of a single round, and one that performs the filtering.

1 Racket

Throws will be represented as the symbols `'r`, `'p` and `'s`.

```
#lang racket
(provide rps)

(define (game-finished? strats) (or (null? strats) (ormap null? strats)))
(define (strats-current strats) (map car strats))
(define (strats-future strats) (map cdr strats))

(define (rps players strategies)
  (if (game-finished? strategies)
    players
    (let* ((current (strats-current strategies))
           (future (strats-future strategies)))
      ; Implement me!
      )
  )
)
```

You can use the `remove-duplicates` function to remove duplicate throws and `sort` with `symbol<?` to order throws.

Example 1

```
; Alice wins in the second round. Charlie loses immediately.
(define players '("alice" "bob" "charlie"))
(define strategies '((r p) (r r) (s p)))

(rps players strategies); '("alice")
```

Example 2 (single round)

```
; Charlie loses because rock beats scissors
(define players '("alice" "bob" "charlie"))
(define strategies '((r) (r) (s)))

(rps players strategies); '("alice" "bob")
```

Example 3 (stalemate, stalemate, win)

```
; First two rounds are stalemates
(define players '("alice" "bob" "charlie"))
(define strategies '((r p r) (p s r) (s r p)))

(rps players strategies) ; '("charlie")
```

2 Haskell

For simplicity, throws will be represented as the characters `'r'`, `'p'` and `'s'`.

```haskell
module Task4 (rps) where
import Data.List

isFinished xs = null xs || any null xs

currentStrategies = map head

futureStrategies = map tail

rps :: [String] -> [[Char]] -> [String]
rps players strategies | isFinished strategies = players
rps players strategies =
    let current = currentStrategies strategies
        future = futureStrategies strategies
    in [] -- Implement me !
```

You can use the `nub` function to remove duplicate throws and `sort` to order the throws.
Example 1

```haskell
-- Alice wins in the second round. Charlie loses immediately.
players = ["alice", "bob", "charlie"]
strategies = [['r', 'p'], ['r', 'r'], ['s', 'p']]

rps players strategies -- ["alice"]
```

Example 2 (single round)

```haskell
-- Charlie loses because rock beats scissors
players = ["alice", "bob", "charlie"]
strategies = [['r'], ['r'], ['s']]

rps players strategies -- ["alice", "bob"]
```

Example 3 (stalemate, stalemate, win)

```haskell
-- First two rounds are stalemates
players = ["alice", "bob", "charlie"]
strategies = [['r', 'p', 'r'], ['p', 's', 'r'], ['s', 'r', 'p']]

rps players strategies -- ["charlie"]
```