```
;; A player is a structure
;;   (make-player name home wins)
;; where name and home are symbols and wins is a number
(define-struct player (name home wins))

;; A ranking is a (list of player) containing 100 elements
;;    with the players in ascending rank order

;; find-by-rank : ranking  number[<=100] → player
;; Purpose: returns the player with the given rank,
;;               starting from rank 1
(define (find-by-rank a-ranking player-num)
  (local [(define (helper alop at-num)
              (cond [(= at-num player-num) (first alop)]
                    [else (helper (rest alop) (add1 at-num))]))]
    (helper a-ranking 1)))
```

```
(define (find-by-rank a-ranking player-num)
  (cond [(= player-num 1) (first a-ranking)]
        [else (find-by-rank (rest a-ranking) (sub1 player-num))]))
```

```
(define (find-by-rank a-ranking player-num)
  (list-ref a-ranking (sub1 player-num)))
```

```
;; a ranking is a structure
;;   (make-ranking  p1 p2 p3 … p100)
;; where the pi are players
(define-struct ranking p1 p2 p3 … p100)
```

```
;; find-by-rank : ranking  number[<=100] → player
;; Purpose: returns the player with the given rank,
;;              starting from rank 1
 (define (find-by-rank a-ranking player-num)
   (cond          [(= player-num 1) p1]
                  [(= player-num 2) p2]
                  [(= player-num 3) p3]
                  …
                  [(= player-num 100) p100]
    ))
```

```
;; A ranking is a vector of 100 players

;; find-by-rank : ranking number[<=100] → player
;; Purpose: returns the player with the given rank,
;;              starting from rank 1
(define (find-by-rank a-ranking player-num)
  (vector-ref a-ranking (sub1 player-num)))
```

```
;; make-ranking : number → vector
;; Purpose:  creates a vector with all components
;;              initialized to false
(define (make-ranking size)
   (build-vector size (lambda (i) false)))
```

Index starts at 0!

```
;; rank-player! : ranking number player → void
;; Purpose: fill the rank specified by the number argument
;;              with the player argument
;; Effect : changes value of ranking in position rank to player
(define (rank-player! a-ranking rank a-player)
  (vector-set! a-ranking rank a-player))
```

```
(define (scalar* a-num a-vec)
   (build-vector
         (vector-length a-vec)
         (lambda (i) (* s (vector-ref a-vec i)))))
```

```
(define (scalar-arith a-num a-vec an-op)
   (build-vector
         (vector-length a-vec)
         (lambda (i) (an-op s (vector-ref a-vec i)))))
```

```
(define (vector-arith vec1 vec2 an-op)
  (build-vector
      (vector-length vec1)
      (lambda (i) (an-op (vector-ref vec1 i) (vector-ref vec2 i)))))
```