1 Collapsing integer lists (Scheme, 8 points)

Write a function `F` that takes as input a list of integers `lst`, e.g. `'()`, `'(1 1 1 1)`, `'(4 5 4 4 1 1 2 3 1)`, and works recursively as follows:

If the input list is empty, `F` returns the empty list

If the input list contains the same integers, for example `'(1 1 1 1)` or `'(22 22)`, `F` return the list itself as output.

Otherwise, first produce a new list `new_lst` defined by

$$\texttt{new\_lst[k]} = \begin{cases} \min\{\texttt{i} | \texttt{lst[i]} = \texttt{lst[k]},\ \texttt{i} > \texttt{k}\} - \texttt{k} \\ \textit{if } \exists \texttt{i} > \texttt{k} \text{ such that } \texttt{lst[i]} = \texttt{lst[k]} \\ \\ 0 \qquad \textit{otherwise} \end{cases},$$

for $1 \leq$ `k` $\leq$ `length(lst)`.

In other words, to get the `k`-th term of the new list `new_lst`, look at the `k`-th integer in the input list `lst`, call this `N`, and count how many position do you need to switch to get the next occurrence of `N` in `lst`. If the last occurrence of `N` in `lst` is at the `k`-th position, the `k`-th term of `new_lst` is 0.

Then remove all the zeros from `new_lst` and obtain a list `new_new_lst`.
Finally, return `F(new_new_lst)`.

Examples One has

```
(F '(1 1 1 1)) ⟶ '(1 1 1 1)
(F '(22 22)) ⟶ '(22 22)
(F '(4 5 4 4 1 1 2 3 1)) ⟶ '(1)
(F '(1 3 2 1 3 1 2 1)) ⟶ '(1 1)
(F '(0 0 1 1 2 2)) ⟶ '(1 1 1)
(F '(32 100 32 100)) ⟶ '(2 2)
```

because

```
4 5 4 4 1 1 2 3 1 ⟶ 2 0 1 0 1 3 0 0 0 ⟶ 2 1 1 3 ⟶ 0 1 0 0 ⟶ 1
1 3 2 1 3 1 2 1 ⟶ 3 3 4 2 0 2 0 0 ⟶ 3 3 4 2 2 ⟶ 1 0 1 0 ⟶ 1 1
1 1 1 2 1 2 3 1 2 ⟶ 1 1 2 2 3 3 0 0 0 ⟶ 1 1 2 2 3 3 ⟶ 1 0 1 0 1 0 ⟶ 1 1 1
32 100 32 100 ⟶ 2 2 0 0 ⟶ 2 2
```

For testing purposes export the function `F`. So your file should be named `task1.rkt` and start with the following lines:

```
#lang racket
(provide F)
```