Task 1: Storing Sparse Matrix [Scheme, 8 points]

A sparse matrix is a matrix containing a lot of zeros. Usually matrices can be represented in Racket as lists of their rows, e.g. the following matrix

$$\begin{pmatrix} 0 & 3 & 0 & 0 \\ 2 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & -5 & 7 & 0 \end{pmatrix}$$

can be represented in Racket as

```
'((0 3 0 0) (2 0 0 -1) (0 0 0 0) (0 -5 7 0))
```

However, for sparse matrices this representation is inefficient as we store all the zeros. To overcome this issue, one can represent a sparse matrix just by collecting non-zero entries together with their positions. For example, we can represent the matrix above as follows:

```
'((0 1 3) (1 0 2) (1 3 -1) (3 1 -5) (3 2 7))
```

Each triplet `(row col val)` represents a non-zero entry, its row, column, and its value. The rows and columns are indexed from 0. For example, the first triplet expresses that the number 3 is located in the row 0 and the column 1. The second triplet expresses that the number 2 is located in the row 1 and the column 0 etc.

Implementation

Implement a function `(sparse mat)` accepting a matrix `mat` represented as a list of rows and creates its sparse representation as above.
Make sure your file is called `task1.rkt` and starts with:

```
#lang racket
(provide sparse)
```

Example 1:

```
> (sparse '((0 3 0 0) (2 0 0 -1) (0 0 0 0) (0 -5 7 0)))
'((0 1 3) (1 0 2) (1 3 -1) (3 1 -5) (3 2 7))
```

Example 2:

```
> (sparse '((10 20 0 0 0 0) (0 30 0 40 0 0) (0 0 50 60 70 0) (0 0 0 0 0 80)))
'((0 0 10) (0 1 20) (1 1 30) (1 3 40) (2 2 50) (2 3 60) (2 4 70) (3 5 80))
```