Exercise 2.32.

We can represent a set as a list of distinct elements, and we can represent the set of all subsets of the set as a list of lists. For example, if the set is `(1 2 3)`, then the set of all subsets is `(() (3) (2) (2 3) (1) (1 3) (1 2) (1 2 3))`. Complete the following definition of a procedure that generates the set of subsets of a set and give a clear explanation of why it works:

```
(define (subsets s)
  (if (null? s)
      (list nil)
      (let ((rest (subsets (cdr s))))
        (append rest (map <??> rest)))))
```

Answer.

This program is very similar to the change-counting program of section 1.2.2. The set of all subsets of the set equals

- The set of all but the first subset of the set, joins
- The set of the union of the first subset and the subsequent subsets.

Observe that all the subsets of a set can be divided into two groups: those that do not includes the first subset and those that do. But the latter one is equivalent to a set formed by joining the first subset with the subsequent subsets. And this can be accomplished with a procedure by mapping over the `rest` of the `subsets` of `s`

```
(map (lambda (x)
       (cons (car s) x))
     rest)
```

Thus, we can recursively reduce the problem of generating the set of all subsets of a set to the problem of generating the subsets of each subset of the set and then merge them into a set.

Besides, we have to specify the degenerate case:

- If `s` is a null set, all of its subset is an empty list.

We can easily translate this description into a recursive procedure:

```
(define (subset s)
  (if (null? s)
      (list nil)
      (let ((rest (subsets (cdr s))))
        (append rest
                (map (lambda (x) (cons (car s) x))
                     rest)))))
```