Q4: (Tutorial) Warmup: Scheme Lists

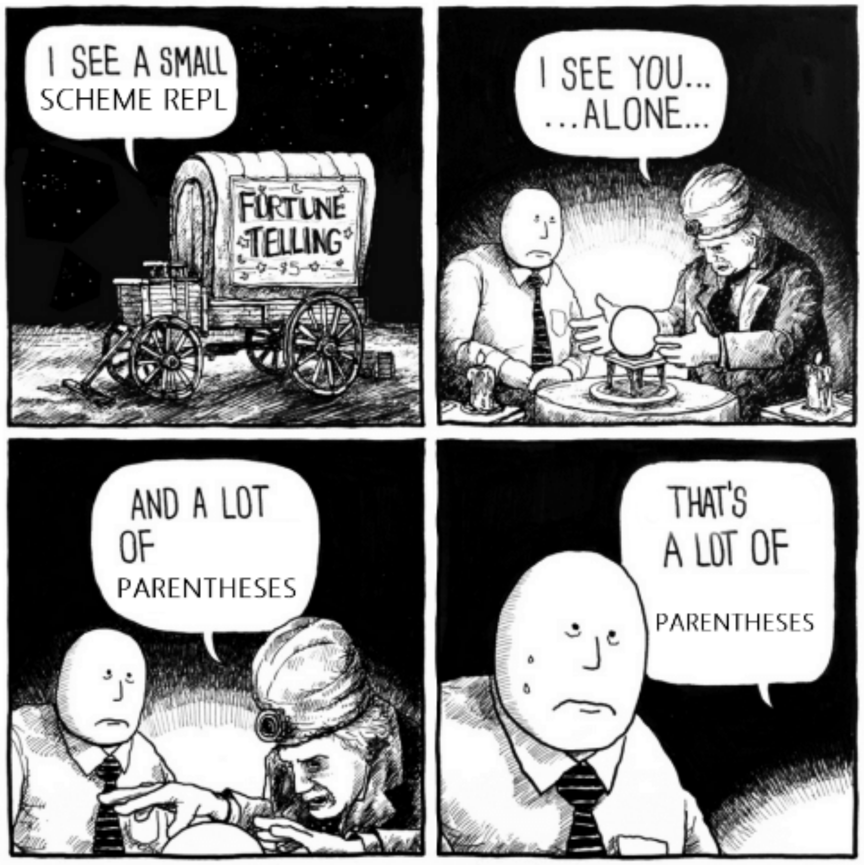
Describe the difference between the following two Scheme expressions. Hint: which defines a new procedure?

```
(define x (+ 1 2 3))
```

```
(define (x) (+ 1 2 3))
```

Write an expression that selects the value 3 from the list below.

```
(define s '(5 4 (1 2) 3 7))
```



Q2: (Tutorial) Fibonacci

Write a function that returns the `n`-th Fibonacci number.

```
; scm> (fib 0)
; 0
; scm> (fib 1)
; 1
; scm> (fib 10)
; 55


(define (fib n)
    'YOUR-CODE-HERE












)
```

Q5: (Tutorial) List Duplicator

Write a Scheme function that, when given a list, such as `(1 2 3 4)`, duplicates every element in the list (i.e. `(1 1 2 2 3 3 4 4)`).

```
(define (duplicate lst)
    'YOUR-CODE-HERE
















)
```

Q6: (Tutorial) List Insert

Write a Scheme function that, when given an element, a list, and an index, inserts the element into the list at that index. You can assume that the index is in bounds for the list.

```
(define (insert element lst index)
    'YOUR-CODE-HERE









)
```