LISTAS EN RACKET

Las listas son parte de nuestra vida diaria. Cuando vamos a mercar, a menudo escribimos una lista de cosas para comprar. Si queremos hacer un plan una mañana, escribimos una lista de las cosas que vamos a hacer. En fin, usar información en forma de listas es una parte normal en nuestras vidas.

Los vectores, a diferencia de las listas, son útiles cuando sabemos cuántos datos deseamos guardar. En muchos casos, no sabemos cuántas cosas deseamos enumerar y que datos guardar, en ese caso creamos una lista. Una lista puede tener un tamaño no determinado y puede contener un conjunto de datos que pueden ser de diversas clases. Las listas son conjuntos que crecen dinámicamente, es decir que se pueden ir añadiendo datos al comienzo o al final de ella de forma simple.

Una desventaja con respecto a los vectores es que no podemos acceder a un dato de forma directa. Por ejemplo, en un vector podemos acceder al dato de la posición 10 diciendo (vector-ref nombre_vector 9), recordemos que empieza en cero. En una lista esto no es posible y tendríamos que recorrer la lista de forma recursiva para llegar hasta el dato de la posición 10.

Cuando formamos una lista, siempre empezamos con la lista vacía. En Scheme,

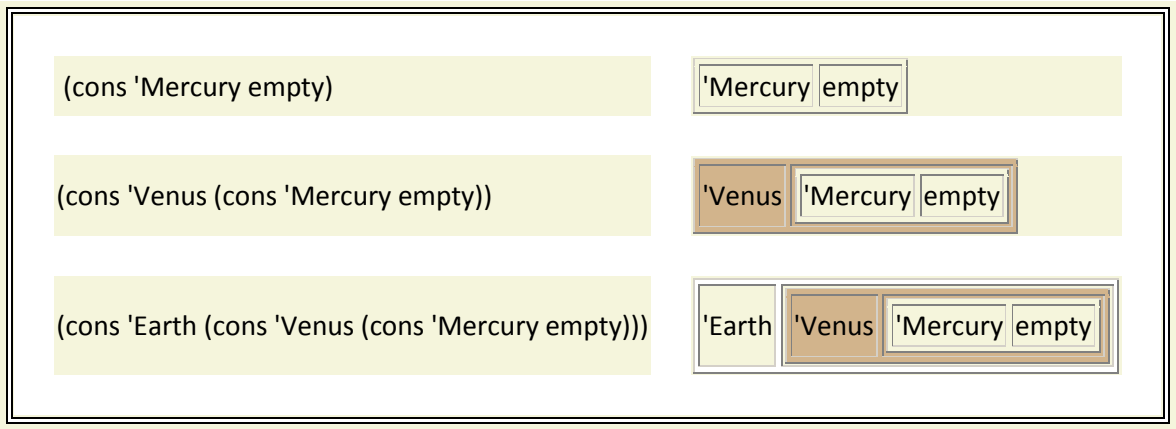
```
empty
```

que representa la lista vacía. Podemos construir una lista mas grande con la operacion `cons`. Aqui hay un ejemplo simple:

```
(cons 'Mercury empty)
```

en este ejemplo se construyó una lista con el símbolo `'Mercury` y la palabra que representa una lista vacía `empty.`

La Figura siguiente presenta esta lista de manera gráfica. La caja que creamos con `cons` tiene dos campos: la `cabeza` y el `resto`. En el primer ejemplo la `cabeza` contiene `'Mercury` y el `resto` contiene `empty`.



Una vez tengamos una lista con un dato, podemos construir listas con dos datos usando `cons` de nuevo:

```
(cons 'Venus (cons 'Mercury empty))
```

La fila de la mitad de la Figura anterior muestra cómo se representa la segunda lista. Es también una caja de 2 campos, pero esta vez el campo `resto` contiene una caja. De hecho, contiene la caja que se creó en la fila de arriba.

Finalmente, construimos una lista con 3 datos:

```
(cons 'Earth (cons 'Venus (cons 'Mercury empty)))
```

En la última fila de la Figura se ilustra la lista con los tres datos. Su campo resto contiene una caja que contiene una caja también. Entonces, al crear listas, estamos poniendo cajas dentro de otras cajas.

Ejercicios. Crear listas que representen

1. La lista de todos los planetas de nuestro sistema solar;
2. Las siguientes comidas: carne, papas-fritas, frijoles, pan, agua, jugo, queso, helado;
3. La lista de los colores básicos.

Haga dibujos de las representaciones de cajas de estas listas, similares a las de la figura anterior.

También podemos hacer una lista de números:

```
(cons 0
 (cons 1
   (cons 2
     (cons 3
       (cons 4
         (cons 5
           (cons 6
             (cons 7
               (cons 8
                 (cons 9 empty)))))))))
```

Para construir esto se requieren 10 construcciones de lista y una lista vacía.
En general una lista no contiene valores de un solo tipo, puede contener valores arbitrarios:

```
(cons 'RobbyRound
 (cons 3
   (cons true
     empty)))
```

Podemos pensar en la lista anterior como un registro personal que incluye el nombre del empleado, el número de años que ha trabajado y si el empleado tiene un plan de salud.

Ejemplo: Enviar una lista de números a una función, sumar los números y devolver el resultado.

```
(cons x (cons y (cons z empty)))
```

Donde `x`, `y`, y `z` son números.

En Scheme existen operaciones para extraer un campo o dato de una lista: first y rest (o, car y cdr). First o car es la operación que permite extraer el primer ítem de la lista (aquel con el que se construyó inicialmente la lista). Rest extrae los demás elementos de la lista.

Los siguientes ejemplos muestran como funcionan first y rest

```
  (first (cons 10 empty))
= 10

  (rest (cons 10 empty))
= empty

  (first (rest (cons 10 (cons 22 empty))))
= (first (cons 22 empty))
= 22
```