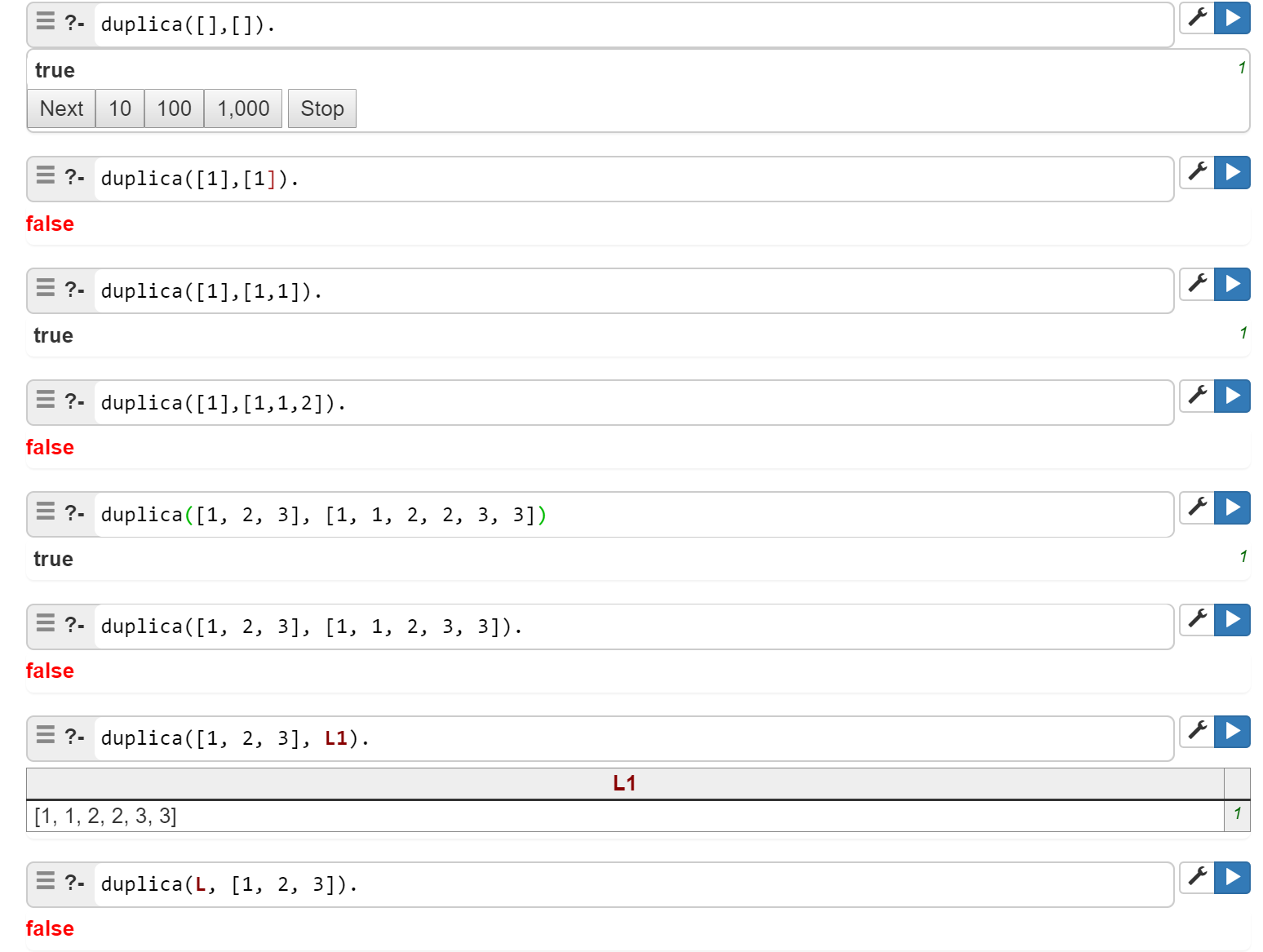
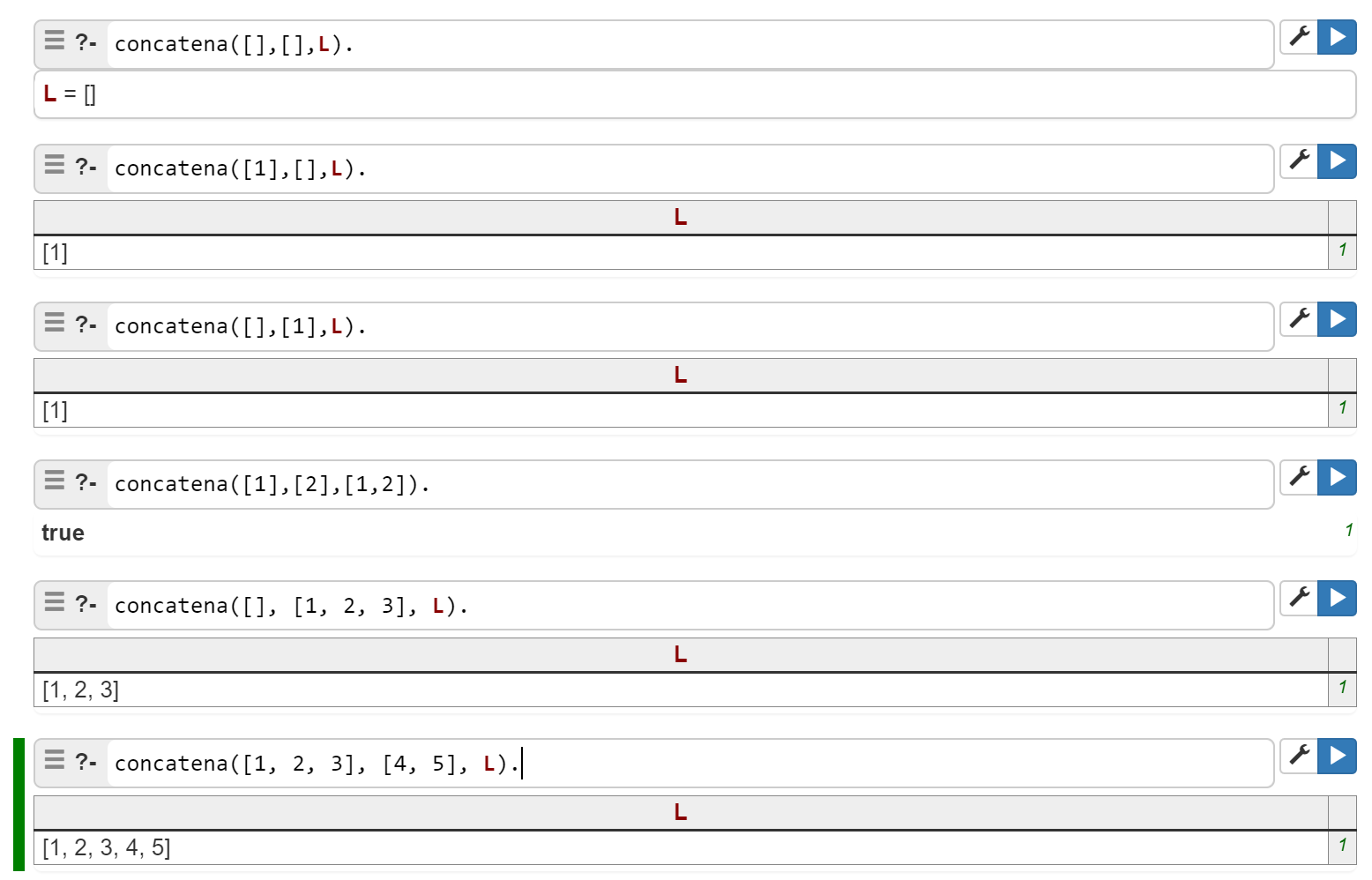
Práctica 3

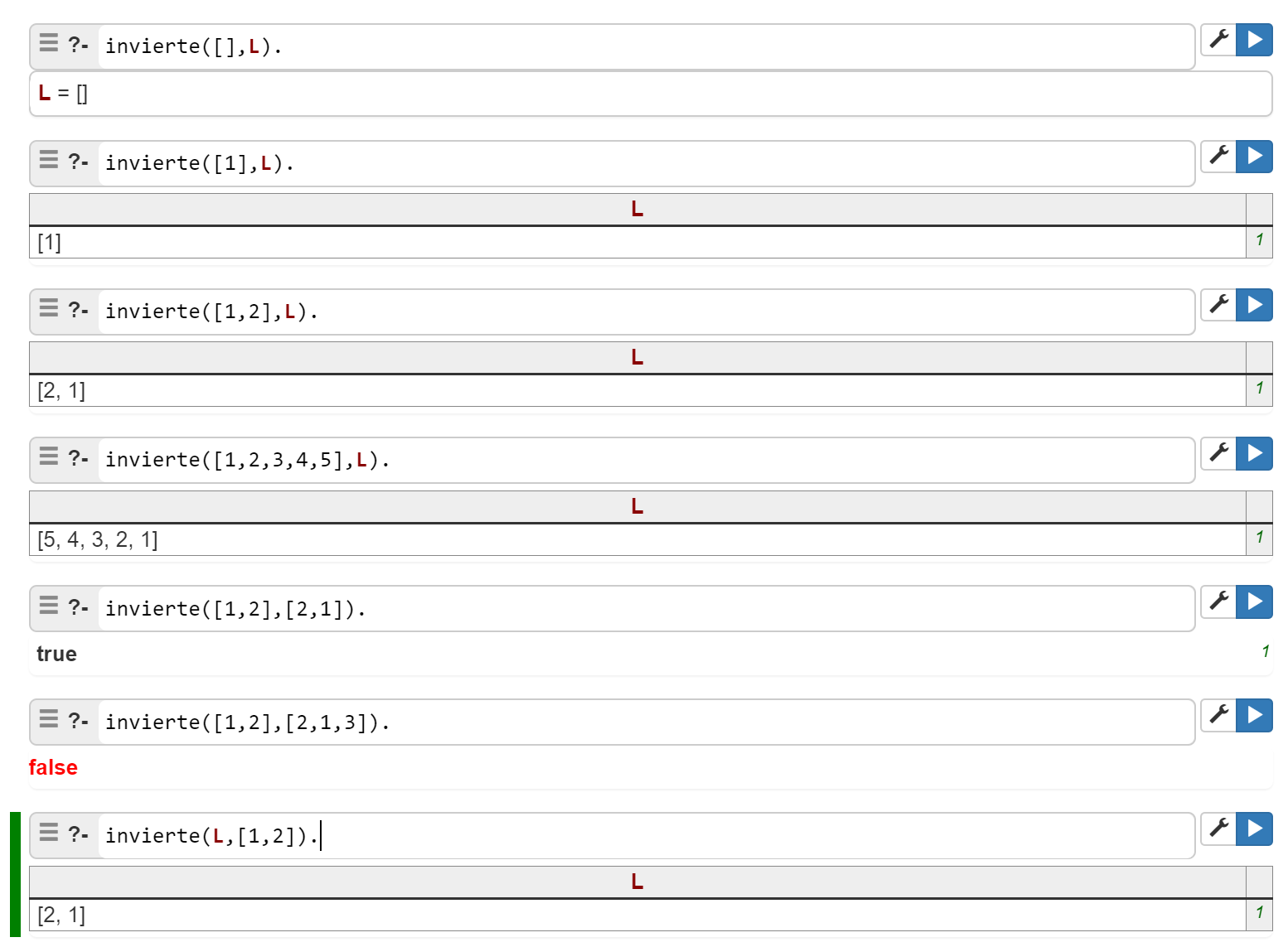
Victoria Pelayo e Ignacio Rabuñal

**Ejercicio 1**

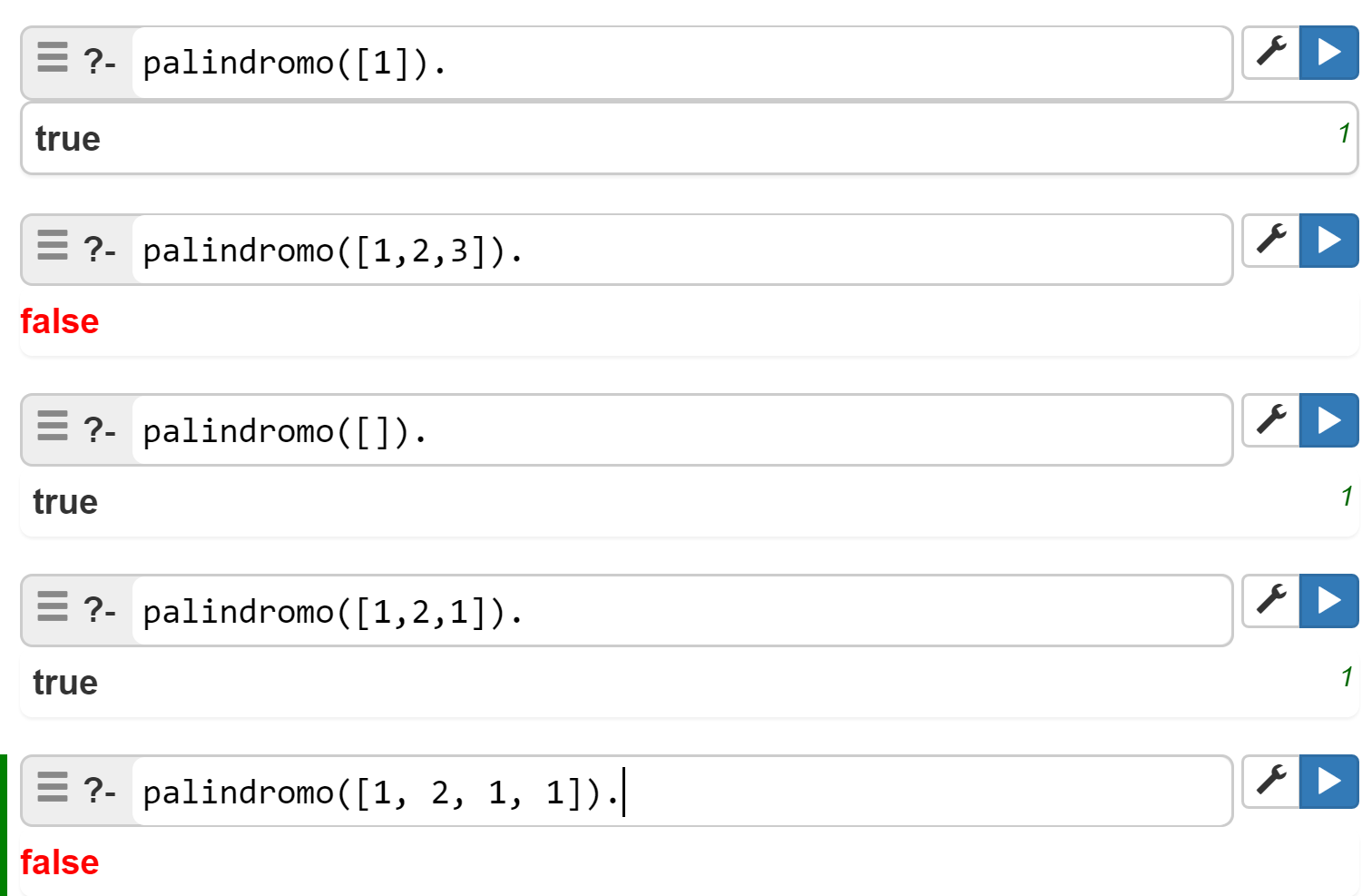


**Ejercicio 2**

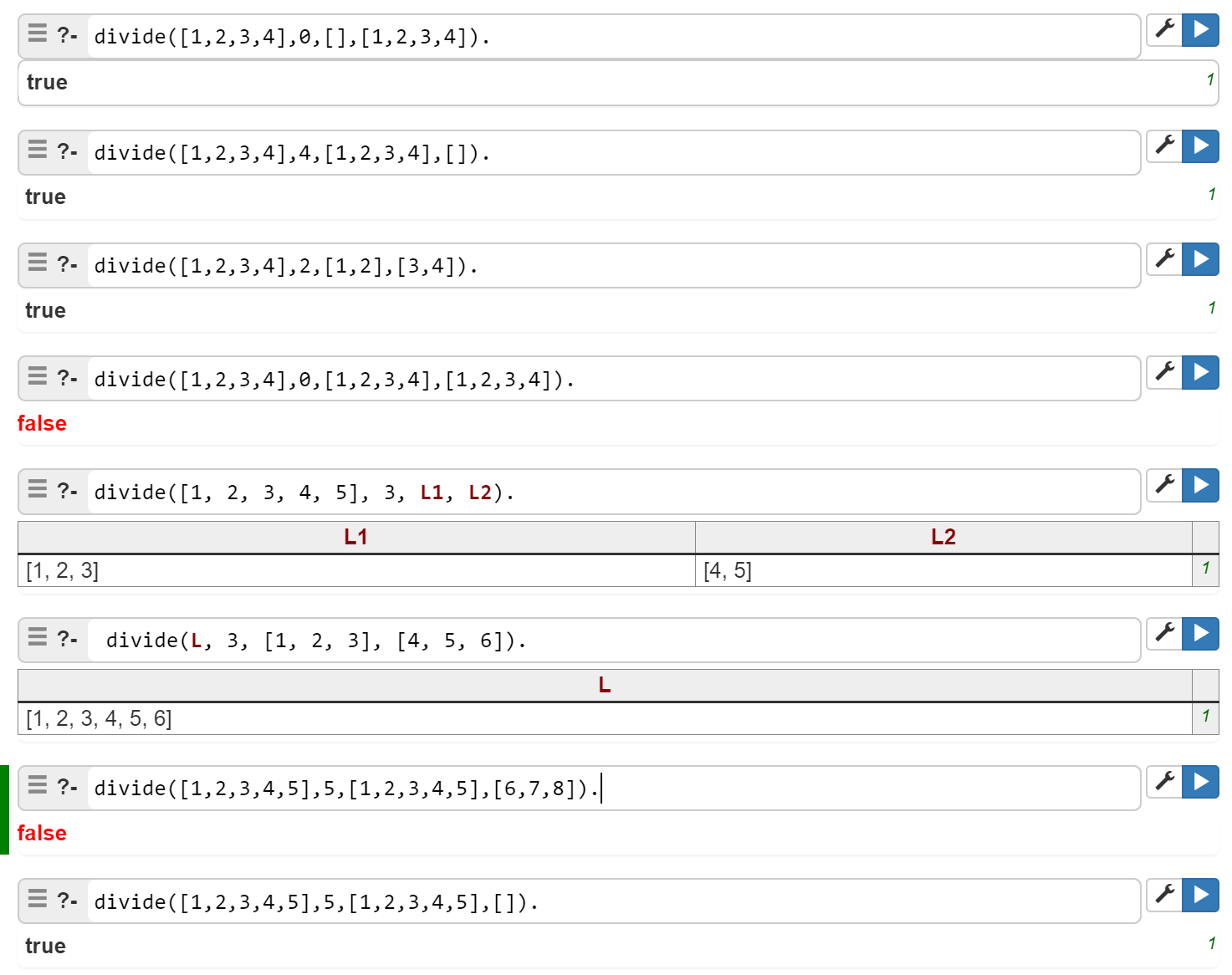




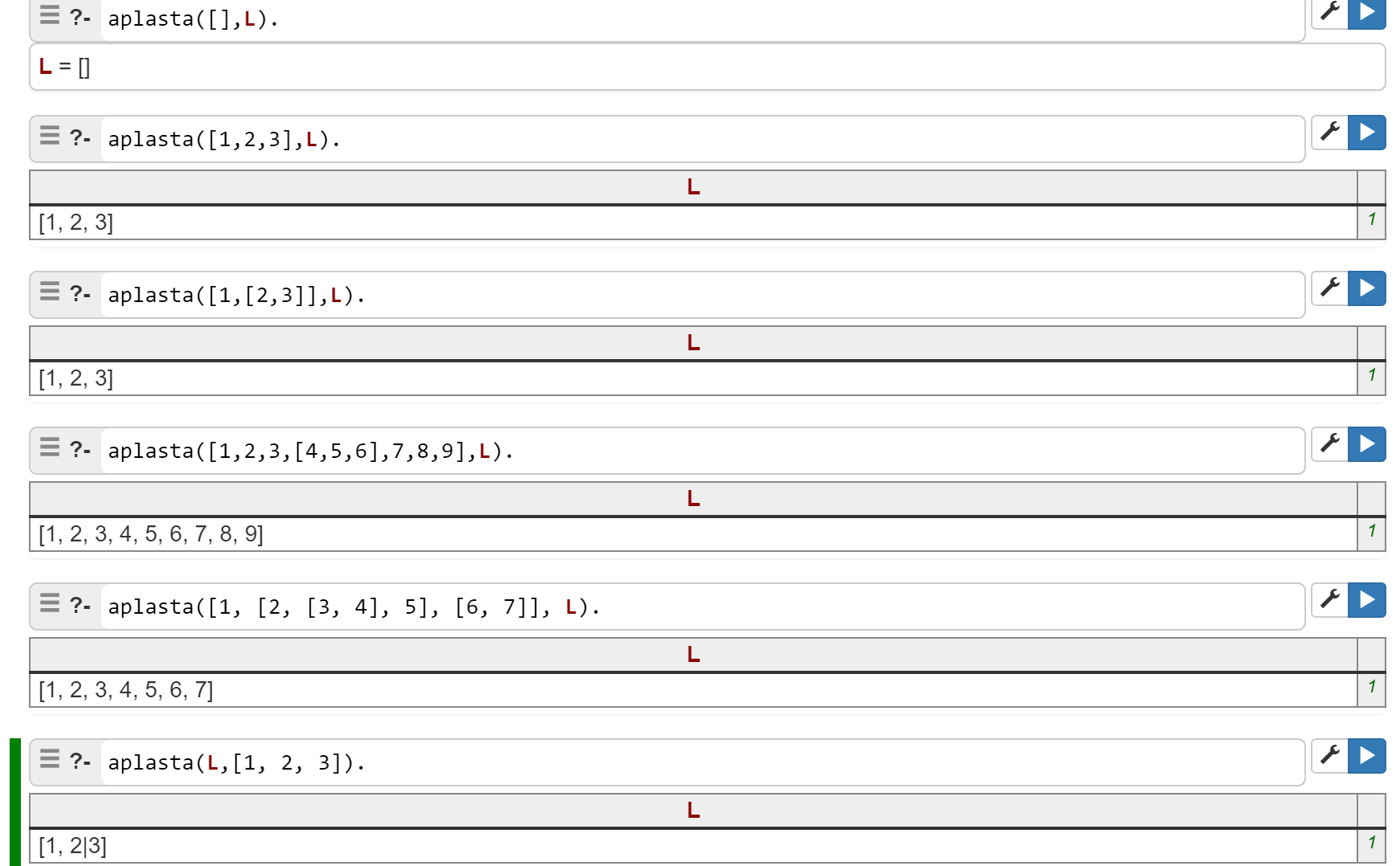
**Ejercicio 3**



**Ejercicio 4**



**Ejercicio 5**



**Ejercicio 6**

**Pseudocódigo:**

Entrada: Número natural del que calcular la factorización (N).

Salida: Lista con los factores primos de N.

Proceso:

Divisor = 2

Mientras N no sea igual a 1:

Si N/Divisor es entero:

N = N/Divisor

Añadir divisor a la lista.

Si no:

Si Divisor == 2:

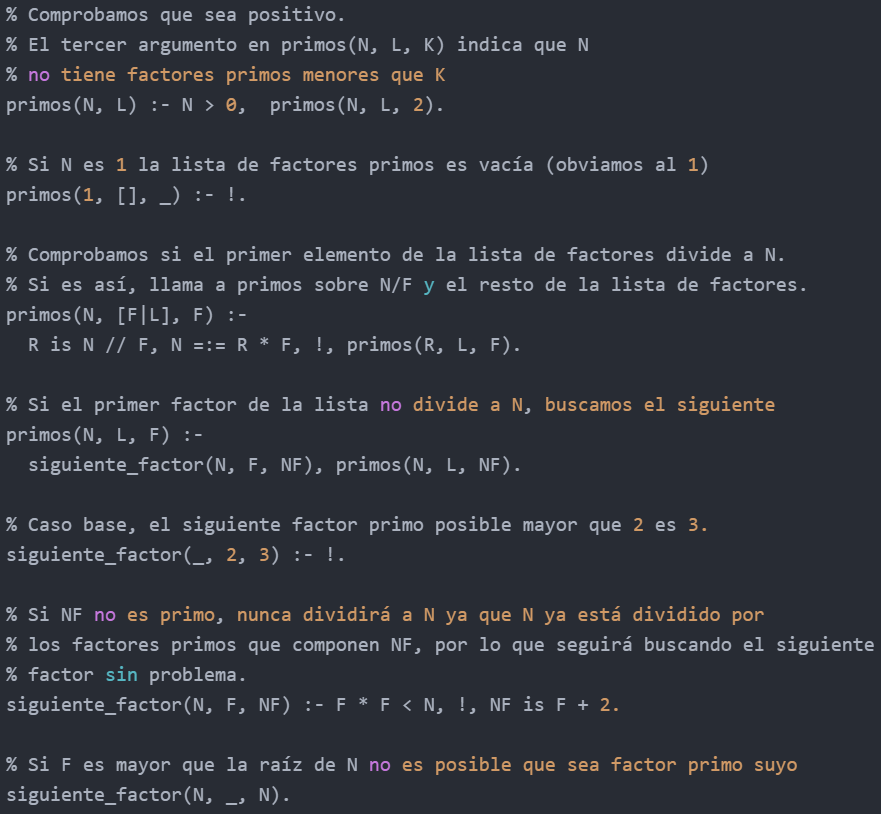
Divisor = 3

Si no:

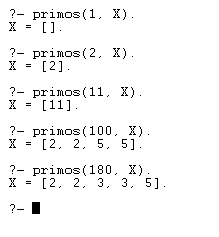
Divisor = Divisor + 2

Devolver la lista.

**Código:**

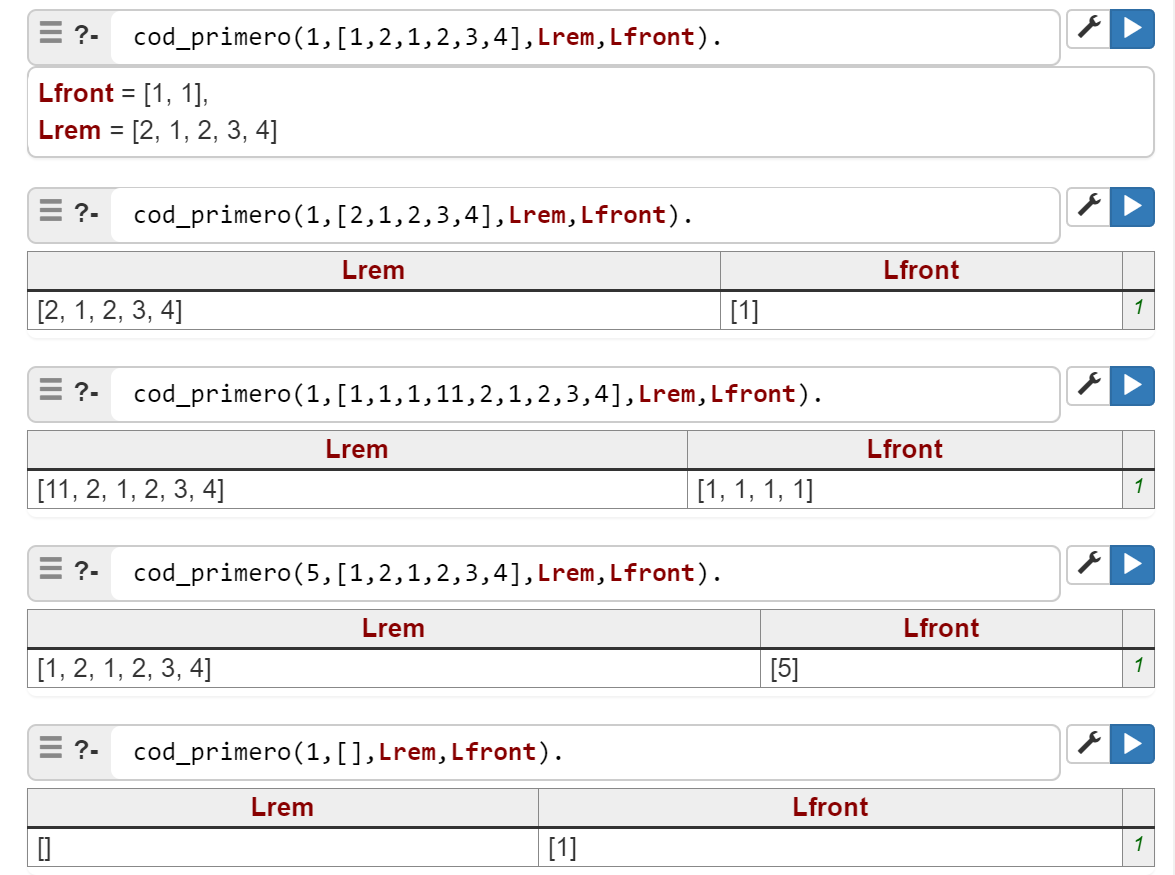


**Ejemplos de ejecución:**

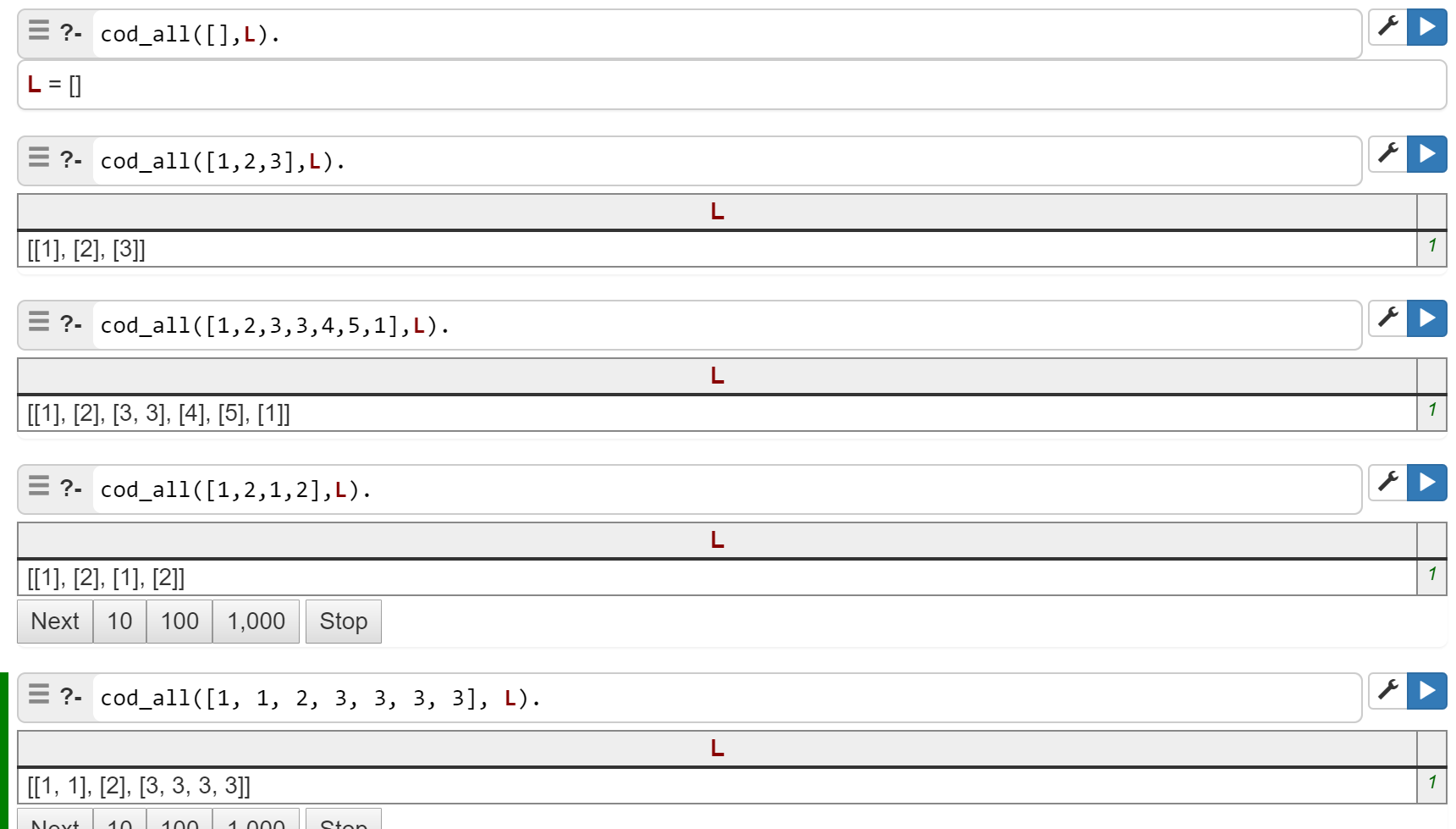


**Ejercicio 7**

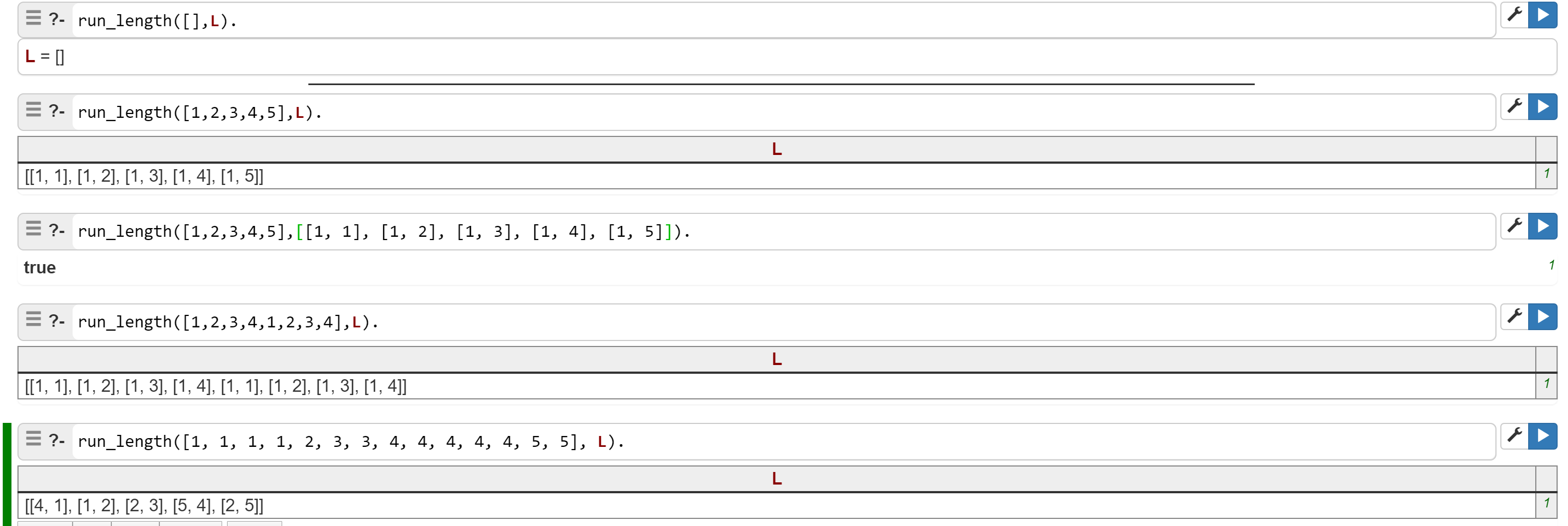
**7.1**



**7.2**



**7.3**



**Ejercicio 8**

**8.0**

**Pseudocódigo:**

Entrada: Lista de elementos a partir de los cuales construir el árbol.

Salida: Árbol de Huffman correspondiente a la lista.

Proceso:

Si la lista tiene sólo un elemento (x):

Devolver tree(x, nil, nil).

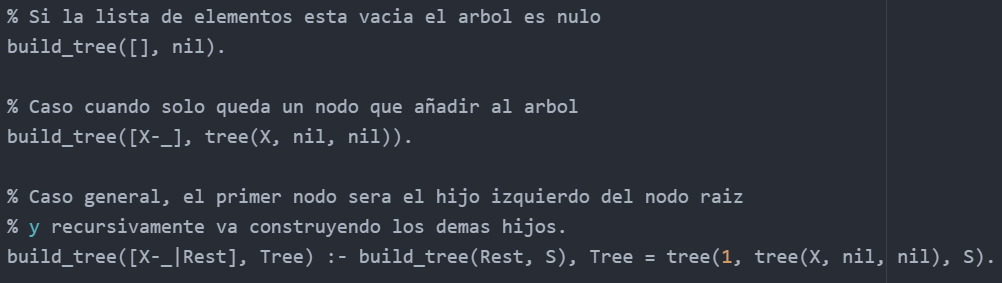
Si no:

X = primer elemento de la lista.

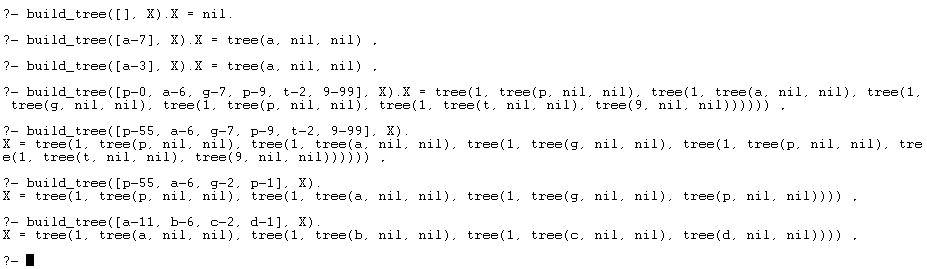
Y = resto de elementos de la lista.

Devolver tree(1, tree(X, nil, nil), build\_tree(Y))

**Código:**



**Ejemplos de ejecución:**



**8.1**

**Pseudocódigo:**

Entrada: Elemento a codificar (X), Árbol de Huffman (Tree).

Salida: Elemento codificado.

Proceso:

N = NodoRaiz(Tree).

Si X es el hijo izquierdo de N:

Devolver [0].

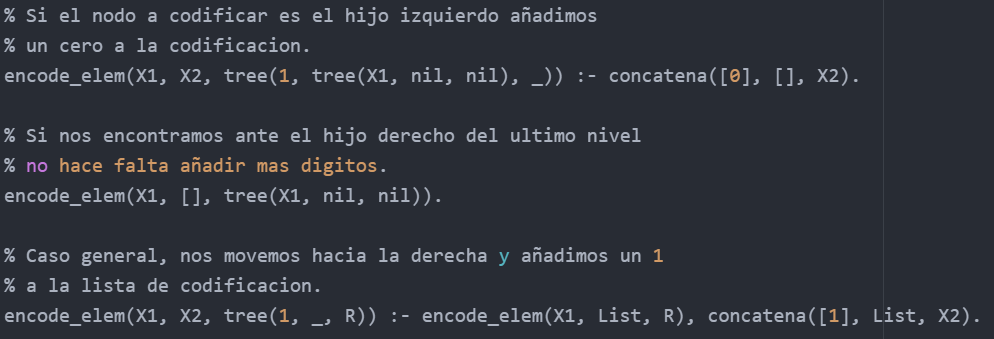
Si X es el hijo derecho de N:

Devolver [].

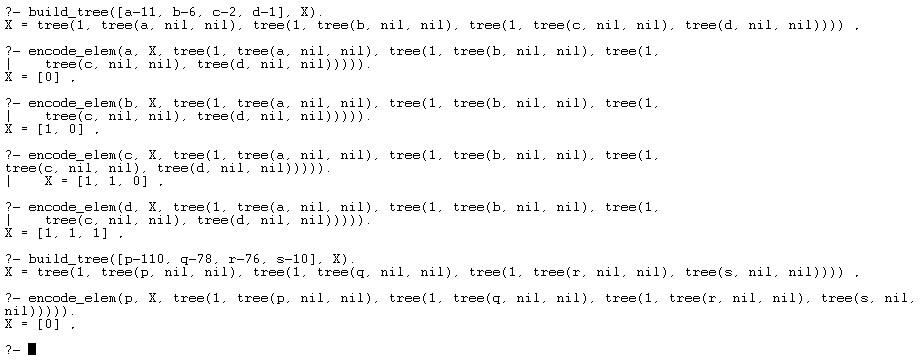
Si no:

Devovler [1, encode\_elem(X, Hijo\_derecho(Tree))]

**Código:**



**Ejemplos de ejecución:**



**8.2**

**Pseudocódigo:**

Entrada: Lista de elementos a codificar (L), árbol usado para codificar (Tree).

Salida: Lista de elementos codificados.

Proceso:

Si solo hay un elemento en L:

X = FIrst(L)

Devolver encode\_elem(X, Tree).

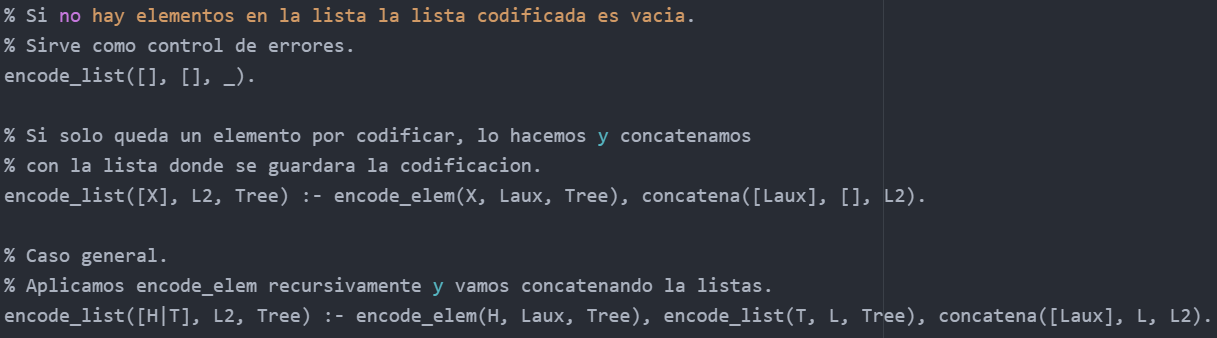
Si hay más:

X = FIrst(L)

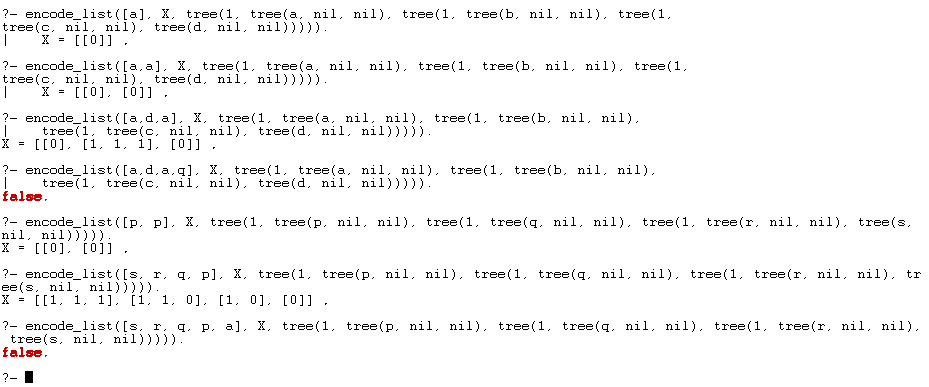
Y = Rest(L)

Devolver [encode\_elem(X, Tree), encode\_list(Y, Tree)]

**Código:**



**Ejemplos de ejecución:**



**8.3**

**Pseudocódigo:**

Entrada: Texto en forma de lista a codificar (L), Diccionario (D).

Salida: Lista con el texto codificado carácter a carácter.

Proceso:

Frecuencias = []

Para cada carácter “c” del diccionario:

F = Contar cuantas veces aparece c en L.

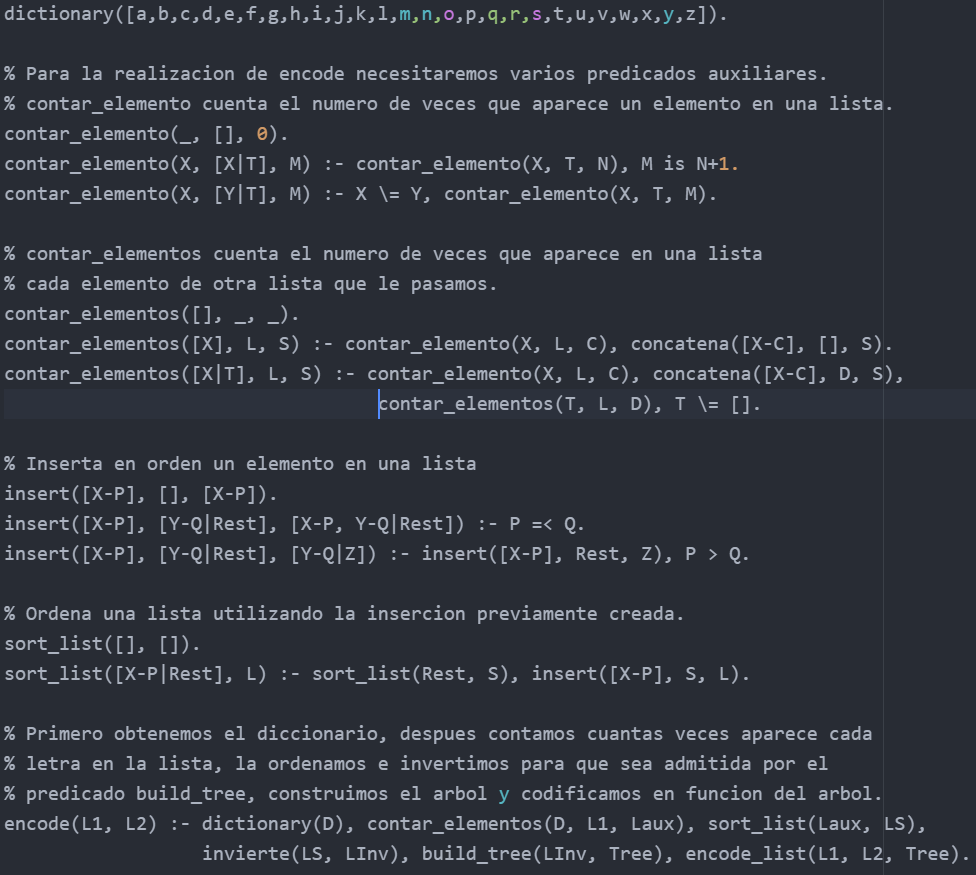
Añadir a frecuencias [c, F].

Ordenar Frecuencias de mayor a menor frecuencia.

Tree= build\_tree(Frecuencias)

Devolver encode\_list(L, Tree)

**Código:**



**Ejemplos de ejecución:**

