Práctica 3

Victoria Pelayo e Ignacio Rabuñal

**Ejercicio 1**

**Pseudocódigo:**

**Duplica(L,L1)**

Input: L(lista), L1(lista duplicada)

Output: True si L1 es la lista duplicada de L

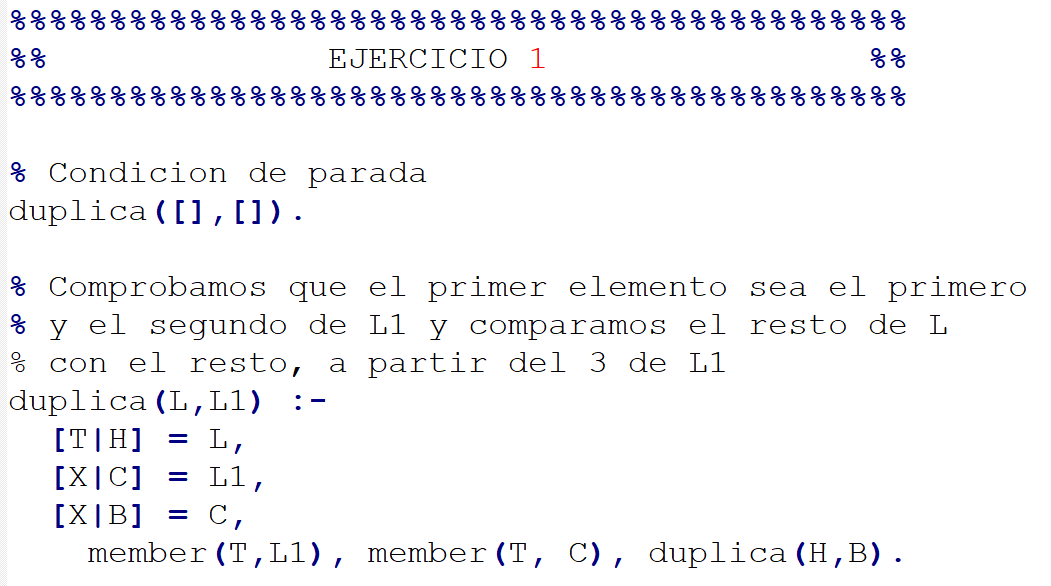
Proceso:

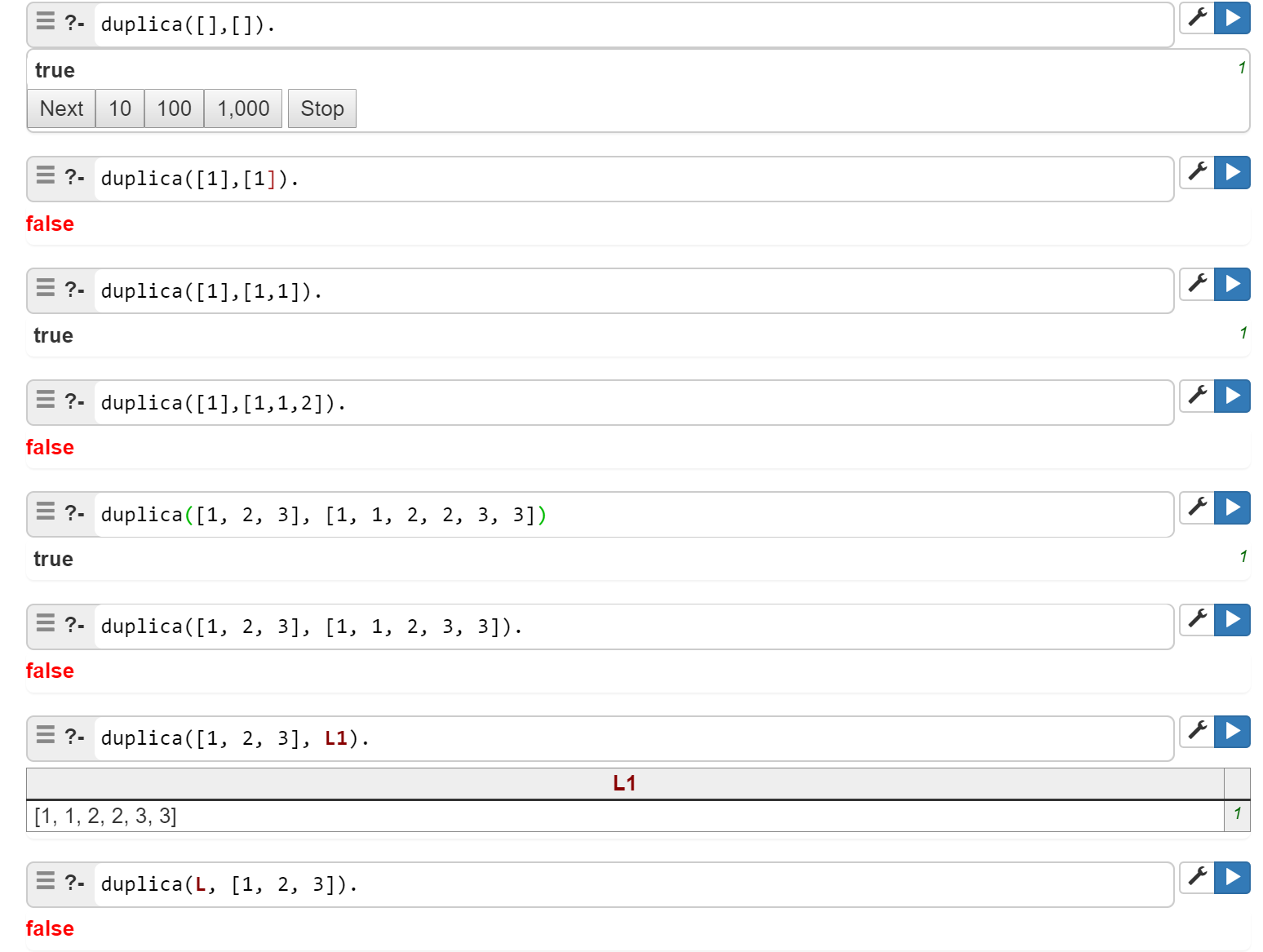
Si L = [] AND L1 = [] devolver true

Primero = L[0]

Devuelvo primero = L1[0] AND primero = L1[1] AND duplica(L[1:],L1[2:])

**Código:**





**Ejercicio 2**

**Pseudocodigo:**

**Invierte(L1,L)**

Input: L(lista para invertir), L1(lista invertida)

Output: True si L1 es la lista invertida de L

Proceso:

Si L = [] AND L1 = [] devolver True

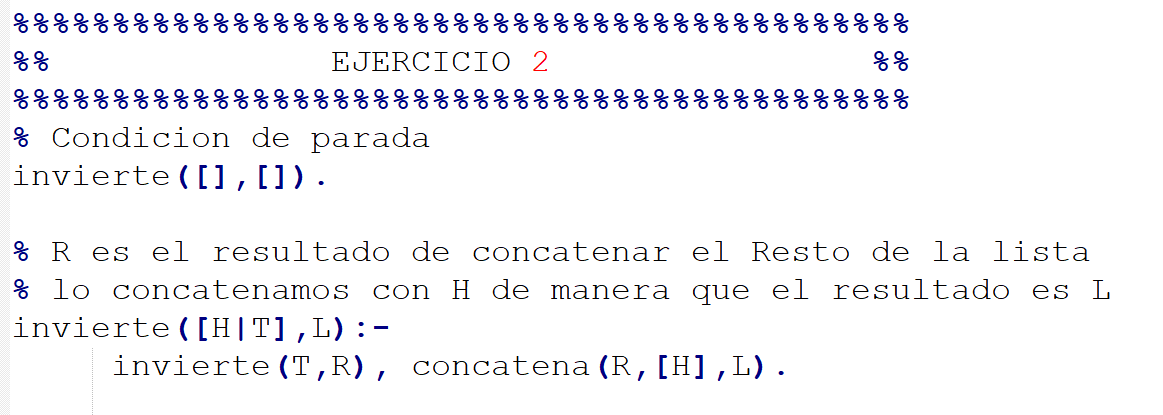
H = L[0]

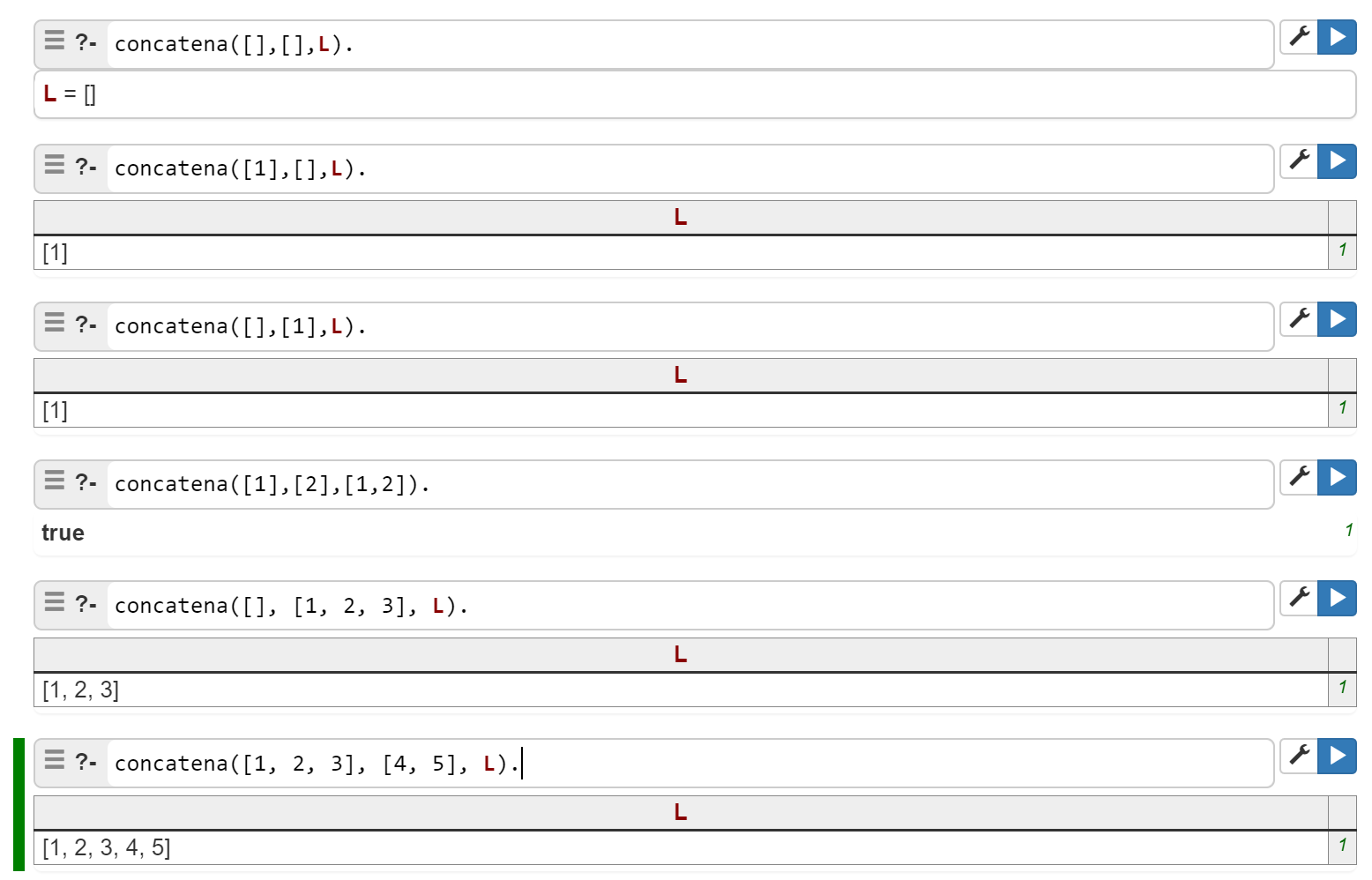
T = L[1:]

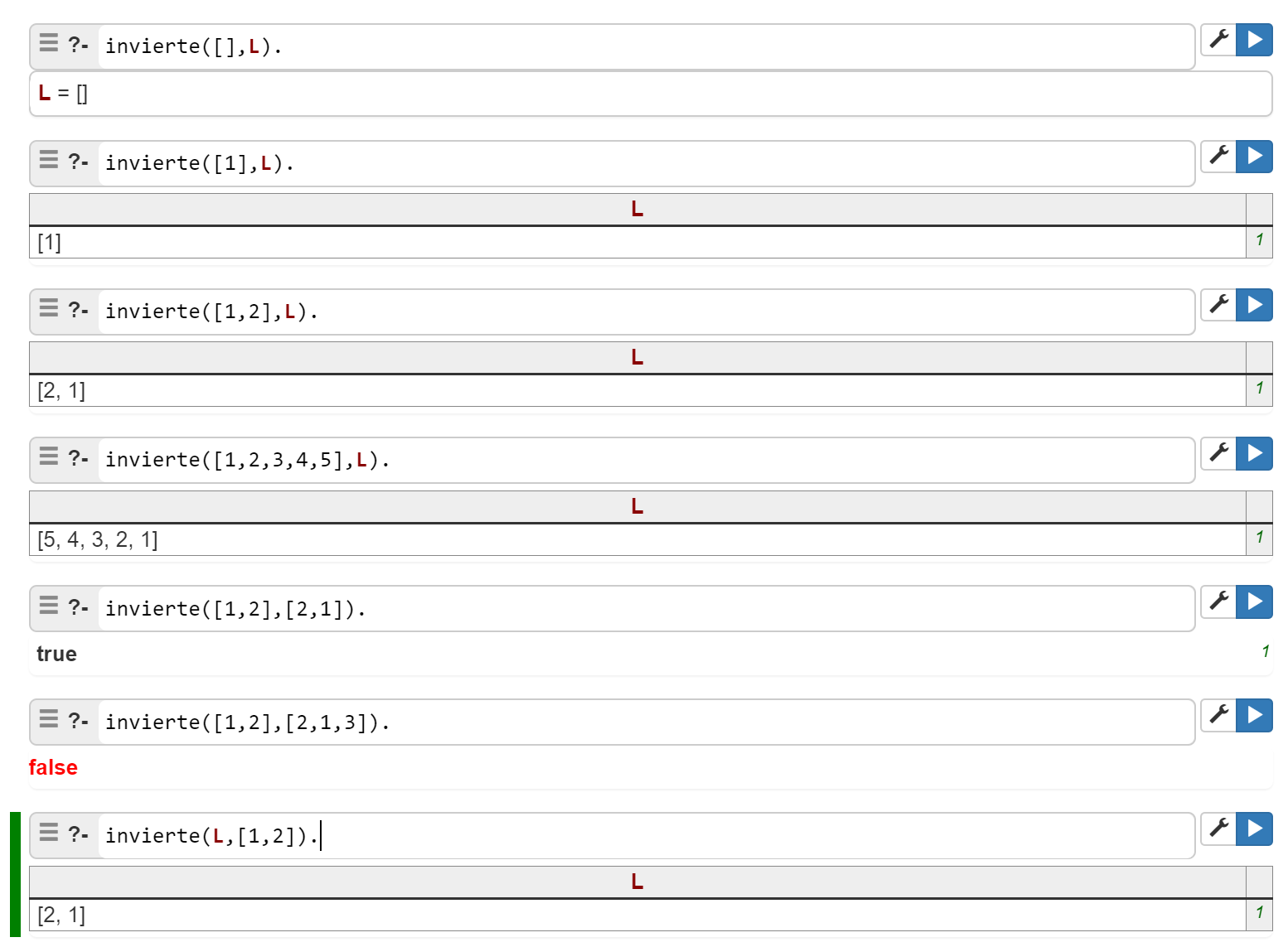
Devolver invierte(T,R) AND concatena (R,[H],L1)

\*concatena(L1,L2,L3): devuelve True cuando L3 es la lista L1 concatenada con L2

**Código:**







**Ejercicio 3**

**Pseudocódigo:**

**Palíndromo(L)**

Input: L(lista)

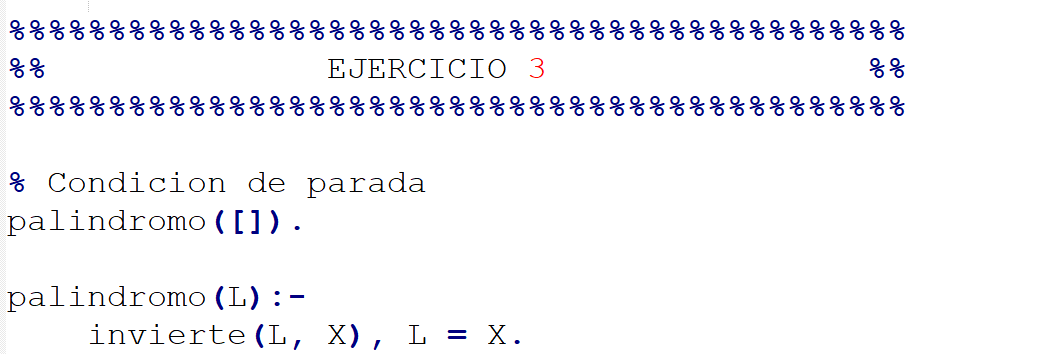
Output: True si L es un palíndromo

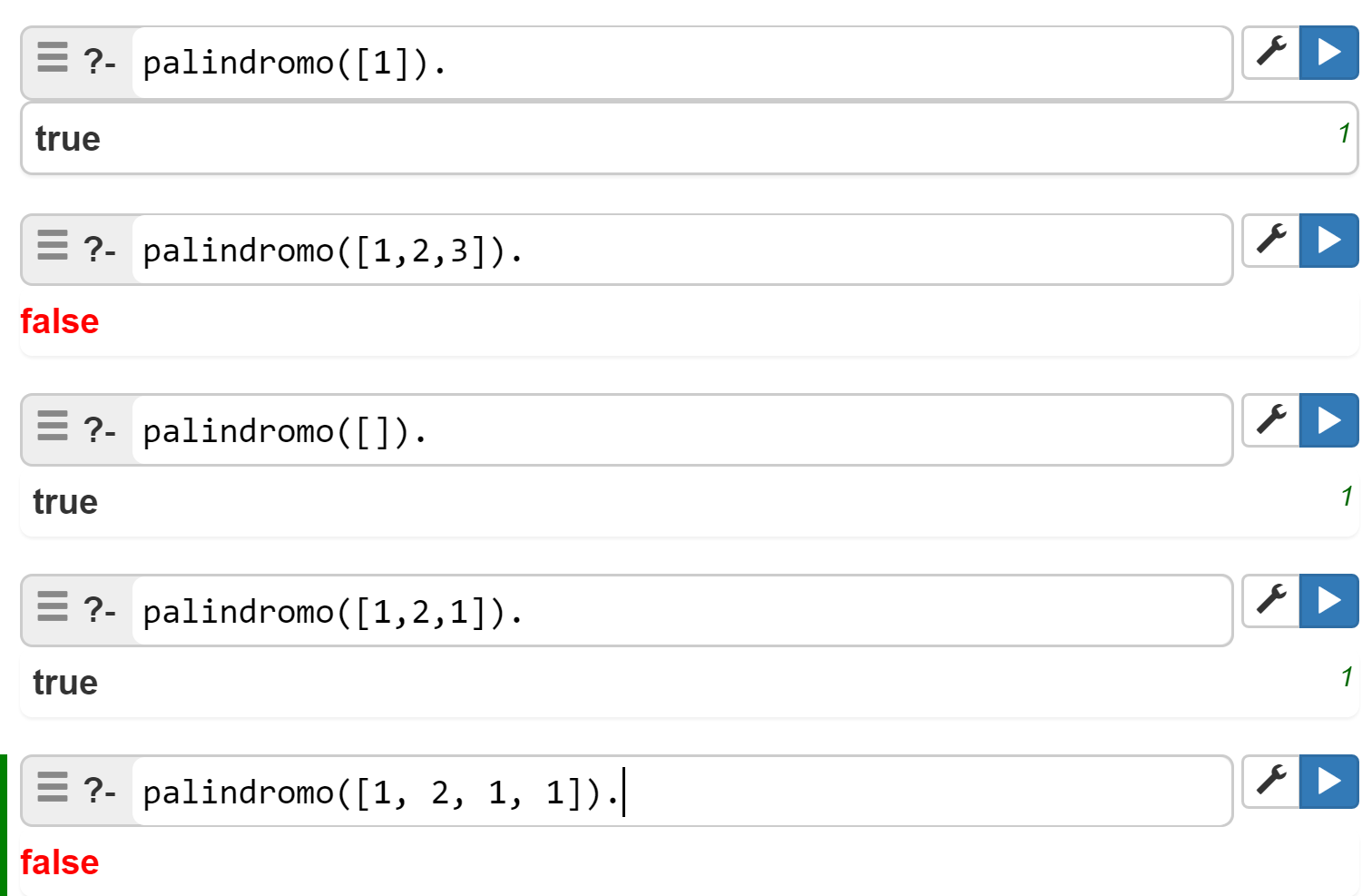
Proceso:

Si L = [] devolver True

Devolver invierte(L,X) AND L = X.

**Código:**





**Ejercicio 4**

**Pseudocódigo:**

**Divide(L,N,L1,L2)**

Input: L, N, L1, L2

Output: Devuelve True si L1 contiene los primeros N elementos de L y L2 los restantes de L

Proceso:

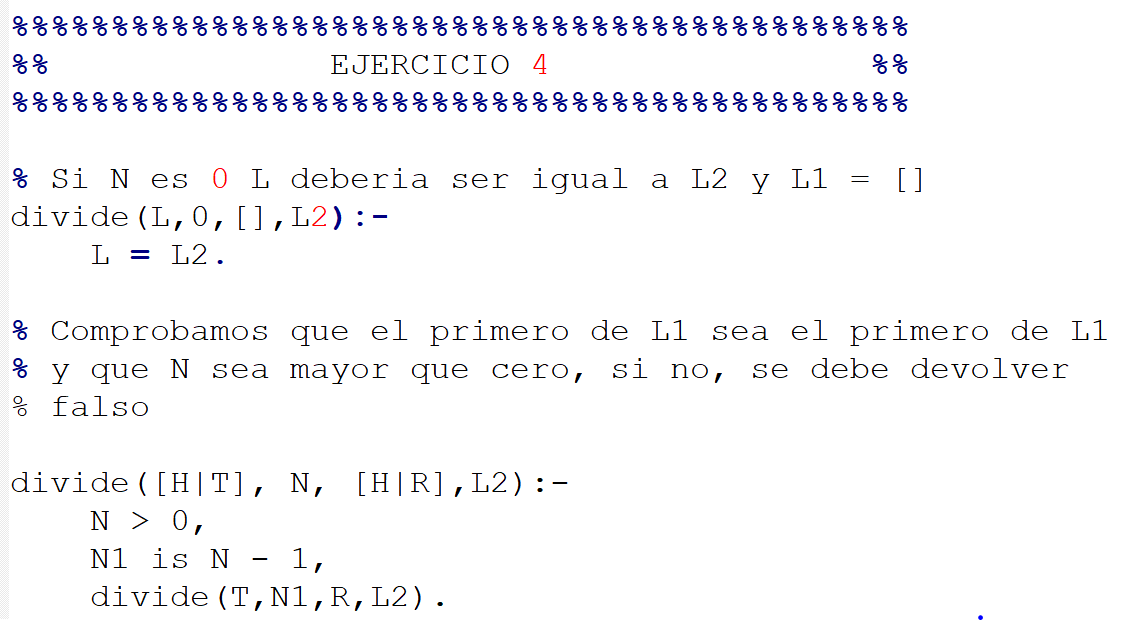
Si N = 0 AND L1 = []:

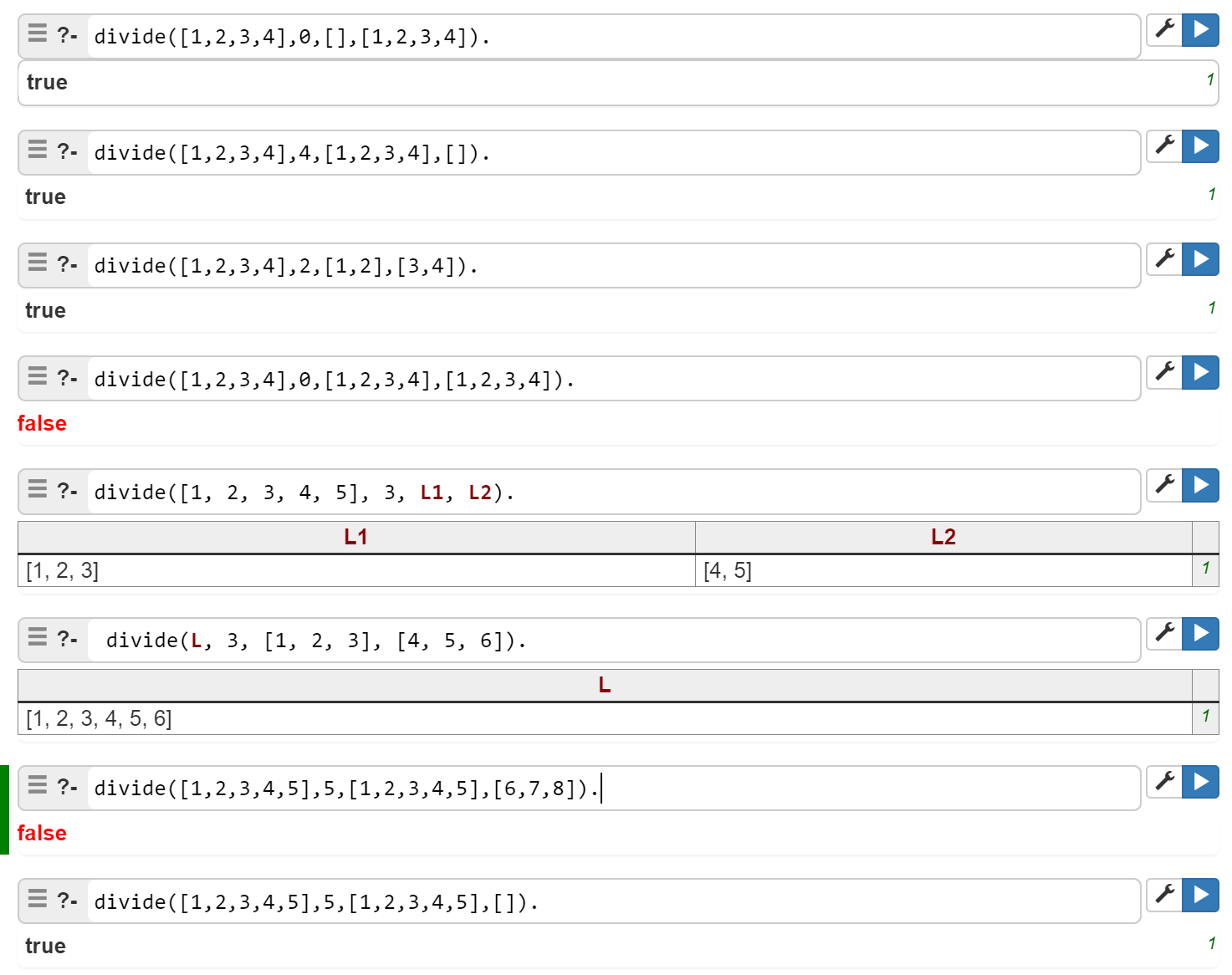
Devolver L = L2 AND True

L1[0] = L[0]

Devolver N > 0 AND N1 = N-1 AND divide(L[1:],N1,L1[1:],L2)

**Código:**





**Ejercicio 5**

**Pseudocódigo:**

**Aplasta(L,L\_aplastada)**

Input: L, L\_aplastada

Output: True si L\_aplastada es la lista aplastada de L

Proceso:

Si L\_aplastada = [L]:

Devolver ¡is\_lista(L) AND aplasta([],[])

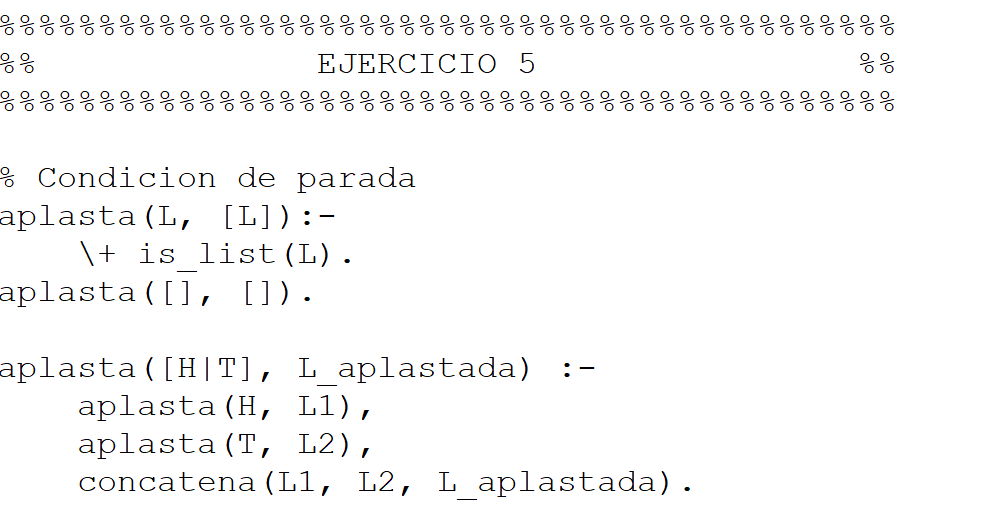
Si L = L\_aplastada = [] devolver True

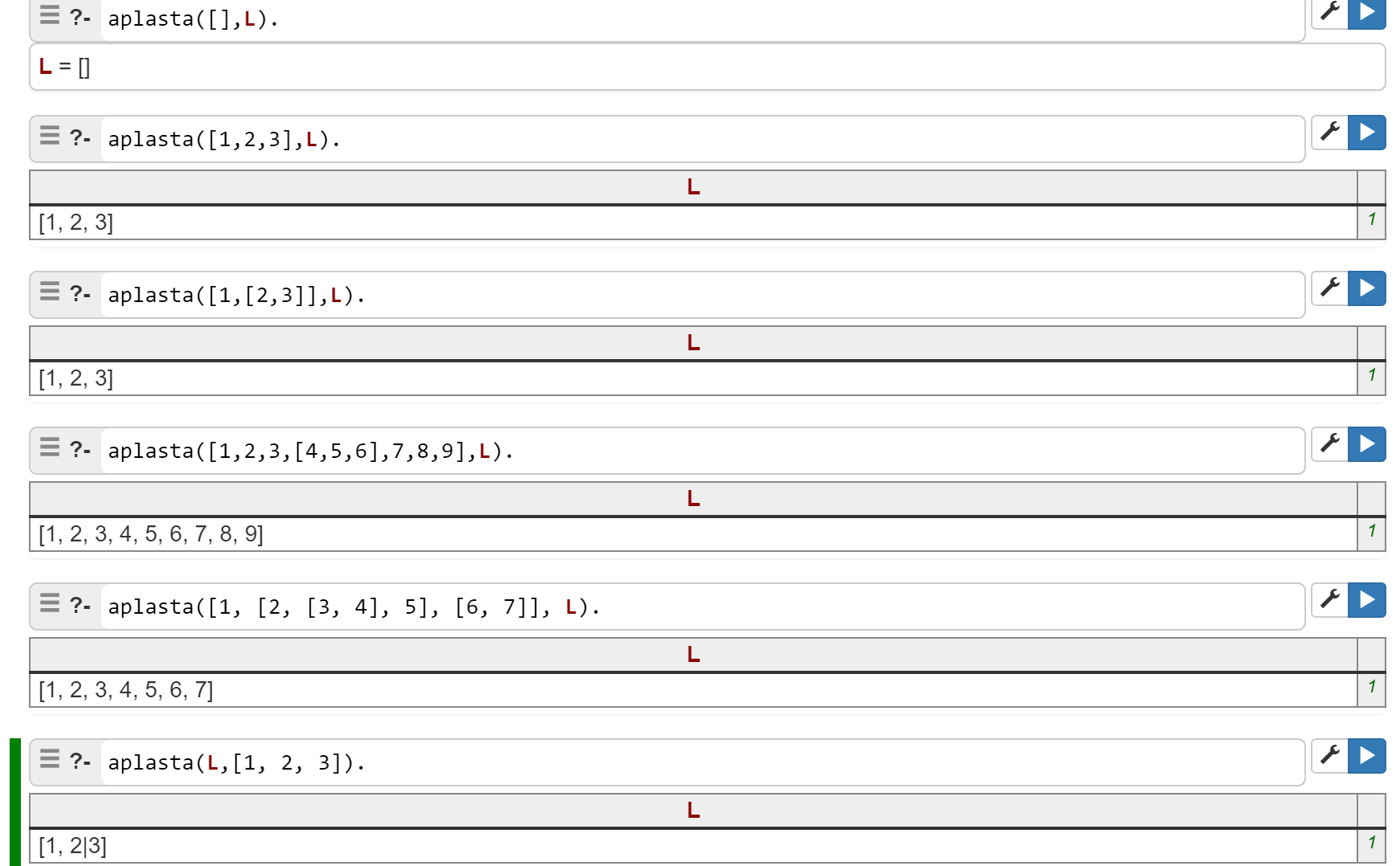
H = L[0]

T = L[1:]

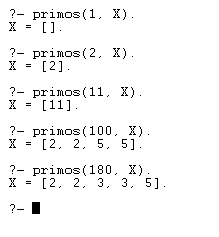
Devovler aplasta(H,L1) AND aplasta(T,L2) AND concatena(L1,L2,L\_aplastada)

**Código:**





**Ejercicio 6**



**Ejercicio 7**

**7.1**

**Pseudocódigo:**

**Cod\_primero(X, L, Lrem, Lfront)**

Input: X(elemento que miramos si coincide), L(lista en la que buscamos L), Lfront(lista con el elemento X + el elemento X el numero de veces que aparece en L, empezando desde el principio), Lrem(resto de la lista L)

Output: Devuelve True si Lfront tiene 1+n el elemento X, siendo n el numero de veces que aparece X empezando desde el principio, y Lrem el resto

**Proceso:**

Si X = L[0]:

Devuleve cod\_primero(X, L[1:], Lrem, L2) AND Lfront = X + L2

Si X != L[0]

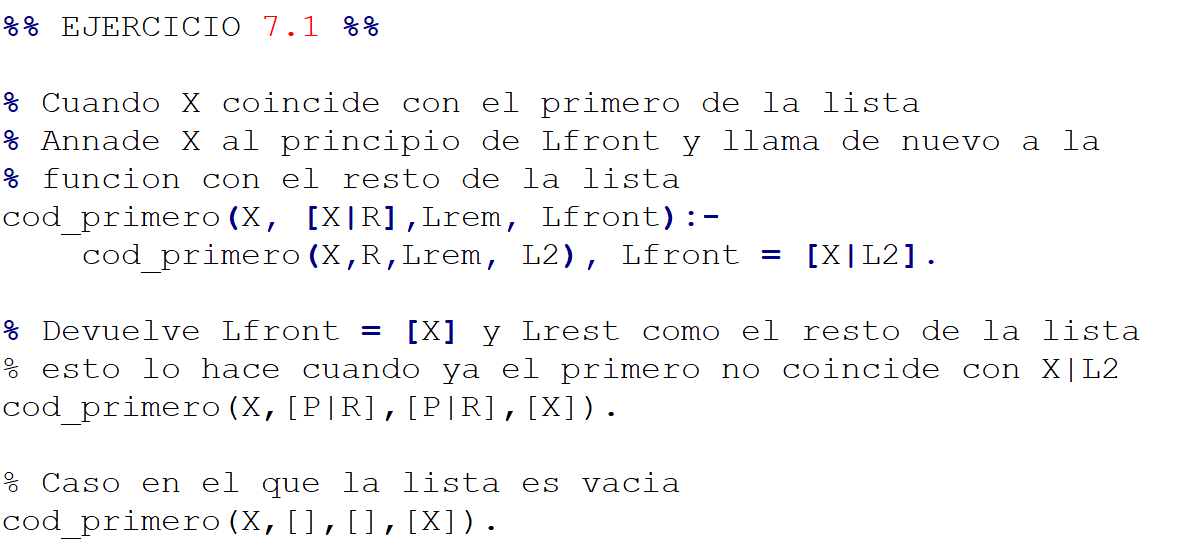
Lrem = L

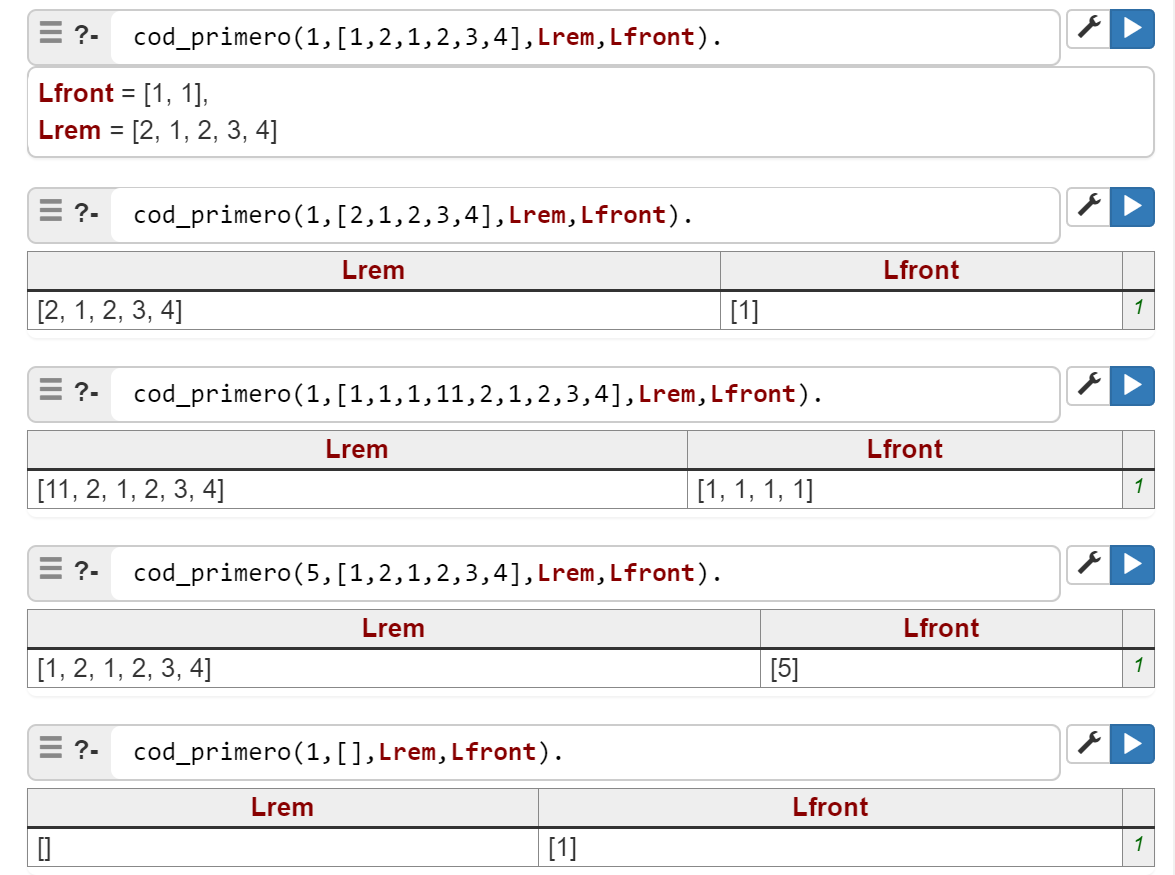
Lfront = [X]

Devuelve True

Si L = [] & Lrem = [] & Lfront = [X] devolver True

**Código:**





**7.2**

**Pseudocódigo:**

**Cod\_all(L, L1)**

Input: L,L1

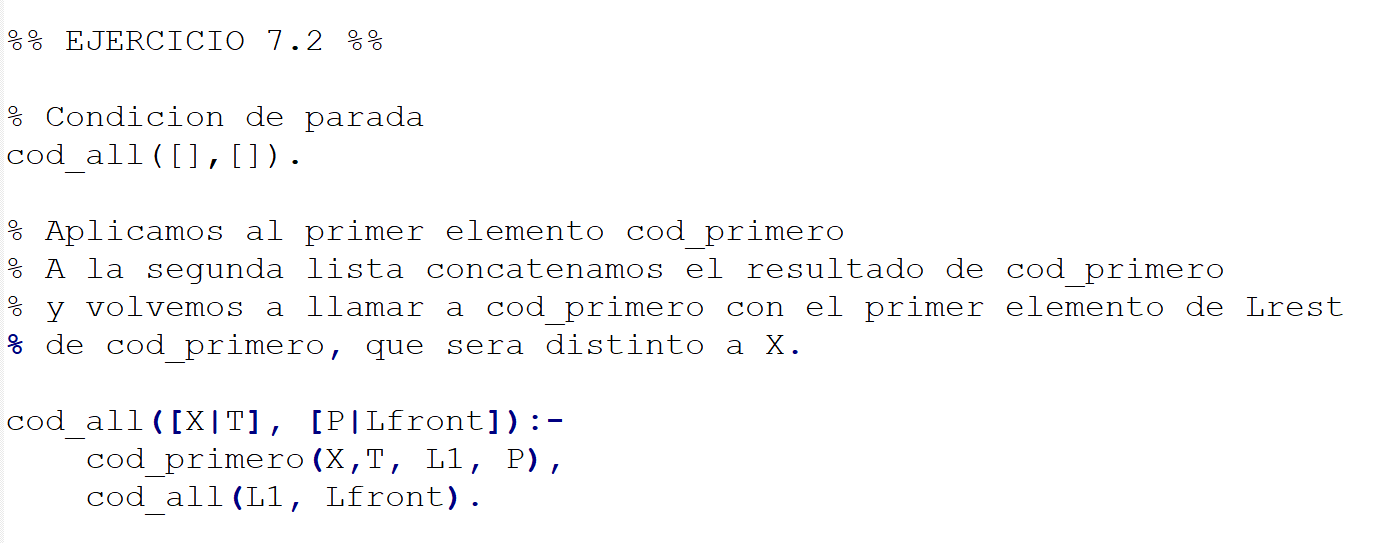
Output: L1 es el resultado de aplicar cod\_primero a todos los elementos de L

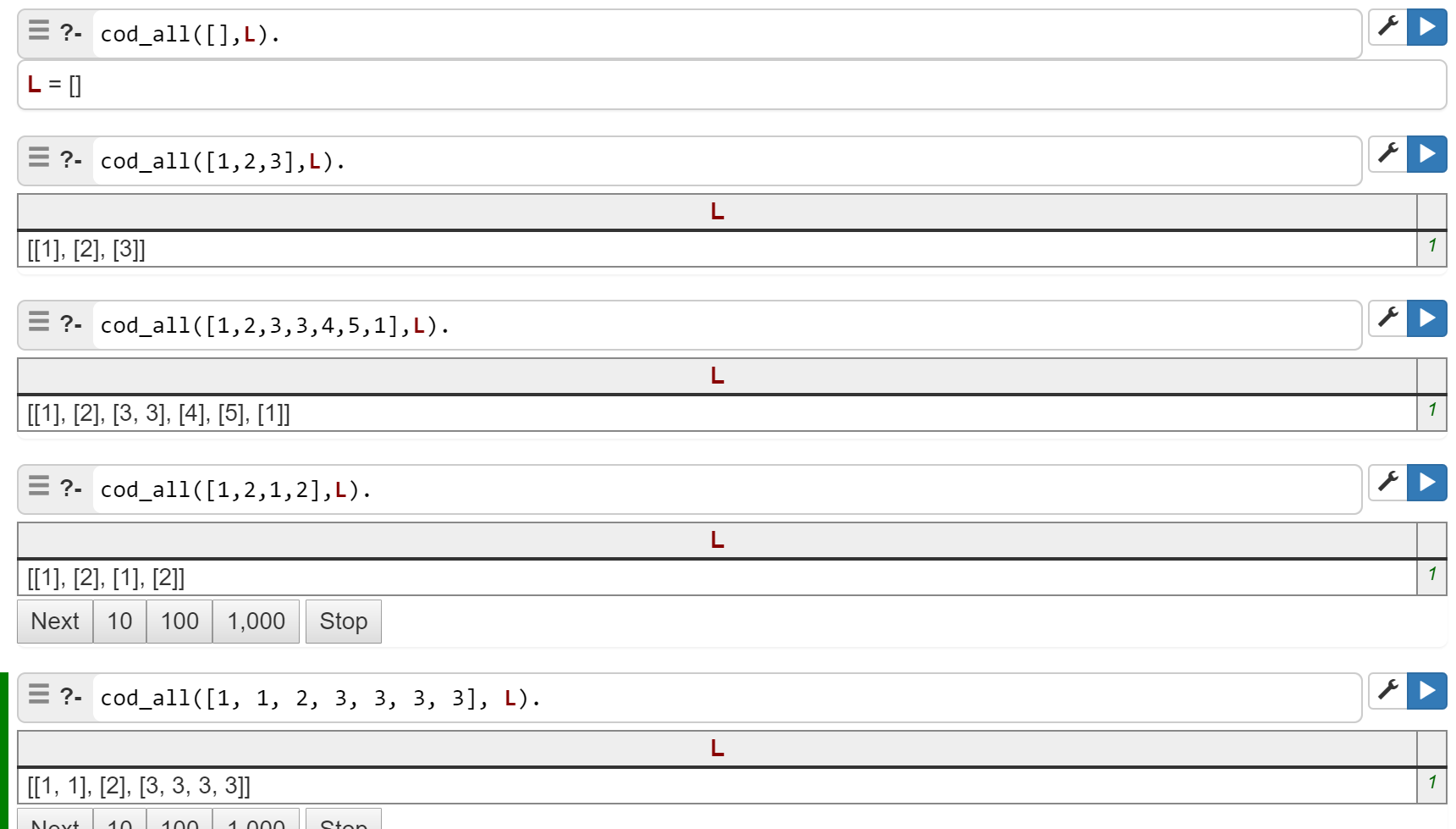
Proceso:

If L = L1 = [] devolver true

Cod\_primero(L[0], L[1:], L’, L1[0]) AND cod\_all(L’, L1[1:])

**Código:**





**7.3**

**Pseudocodigo:**

**Run\_length\_aux**

Input: L,L1

Output: En L1 va concatenando elementos de la forma [X,N], X(lista) elemento de L y N la longitud de X

Proceso:

Si L = L1 = [] devolver True

N = length(L[0])

Devolver run\_length\_aux(L[1:], L1[1:])

**Run\_length**

Input: L1,L2

Output: Devuelve en L2 los elementos con el numero de veces que aparecen seguidas en L1

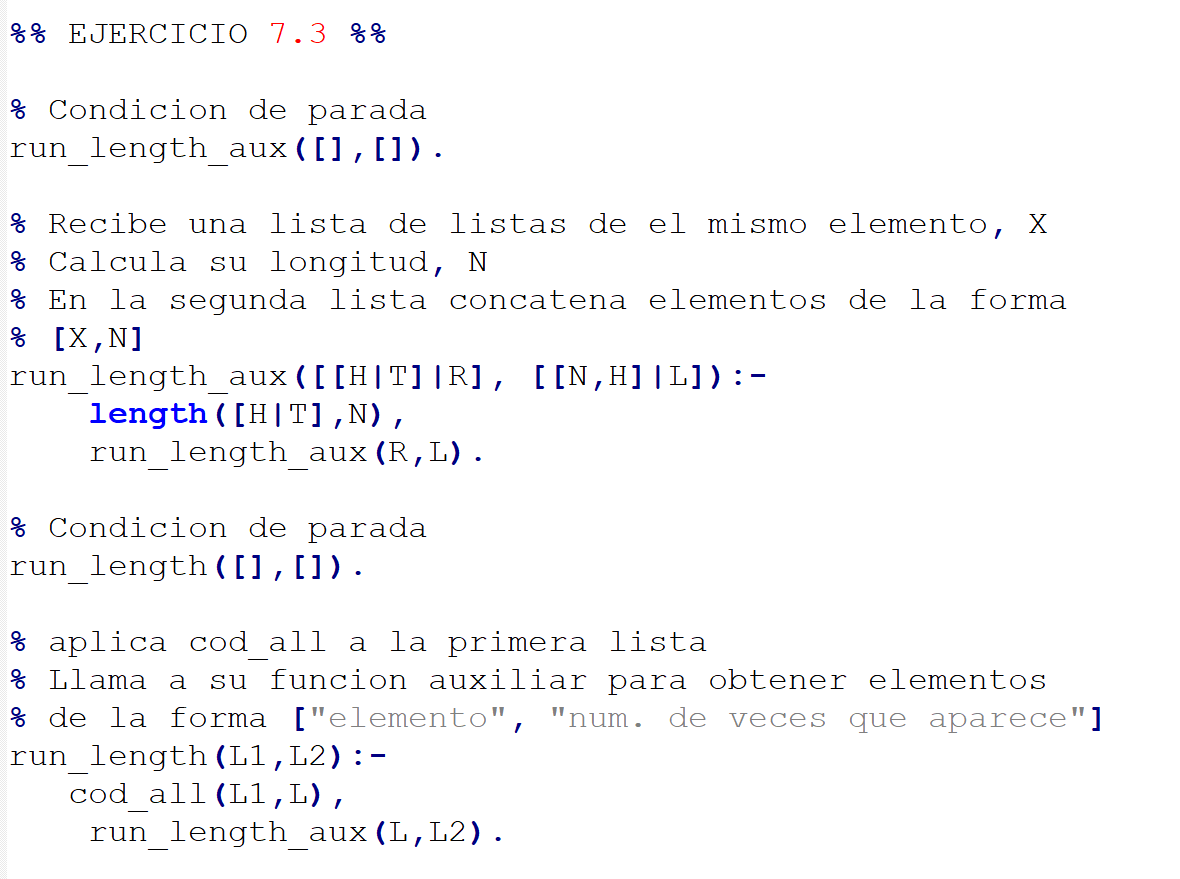
Proceso:

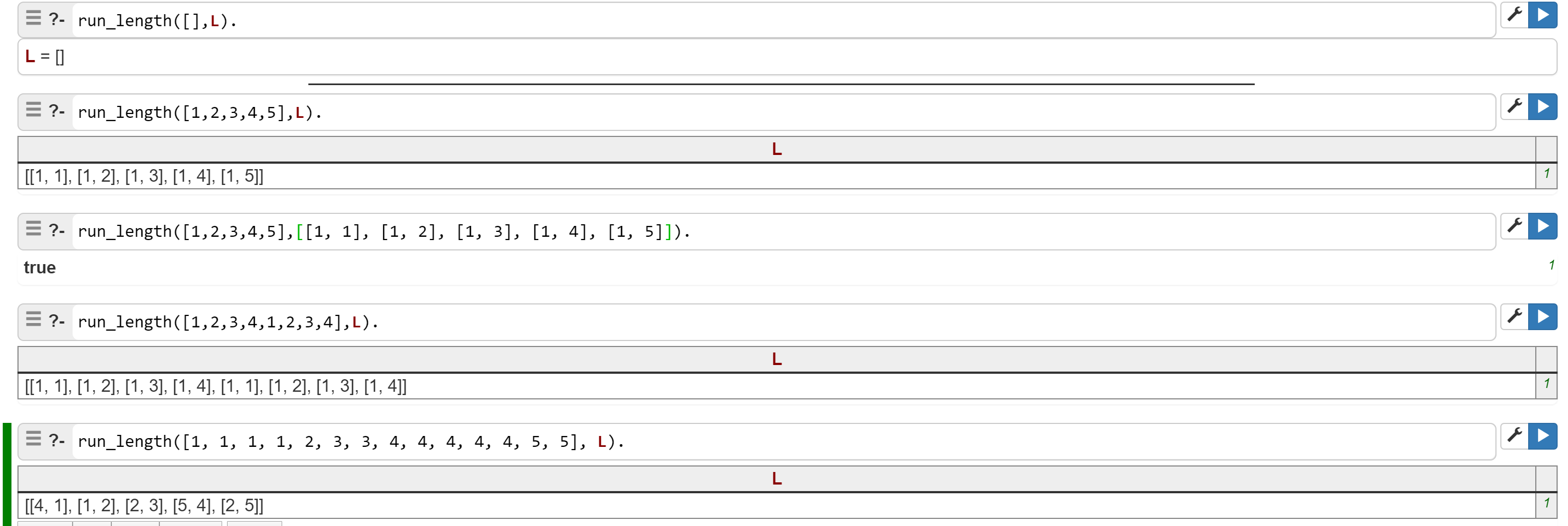
Si L1=l2=[] devolver True

L = cod\_all(L1)

Devolver run\_length\_aux(L, L2)

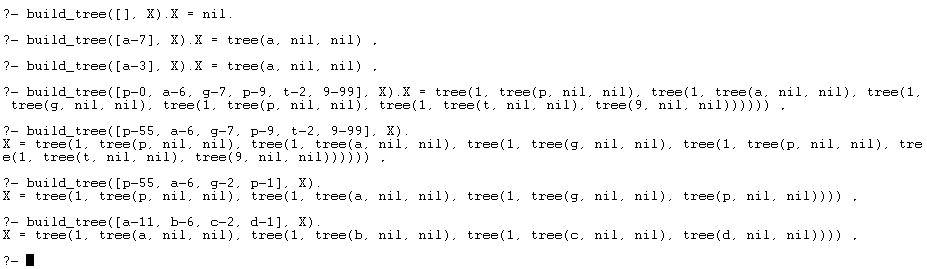
**Código:**



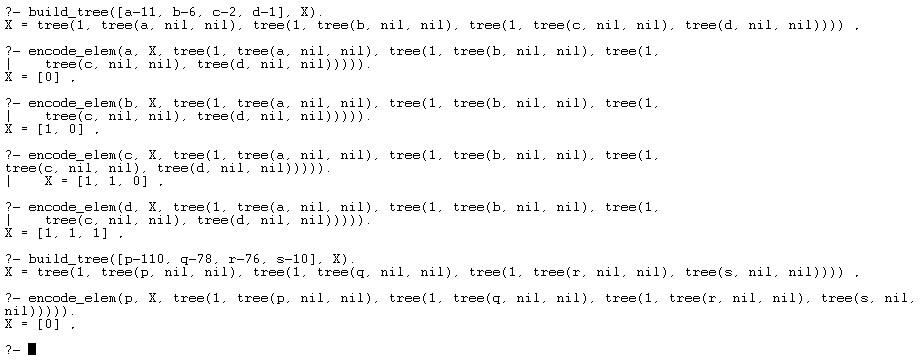


**Ejercicio 8**

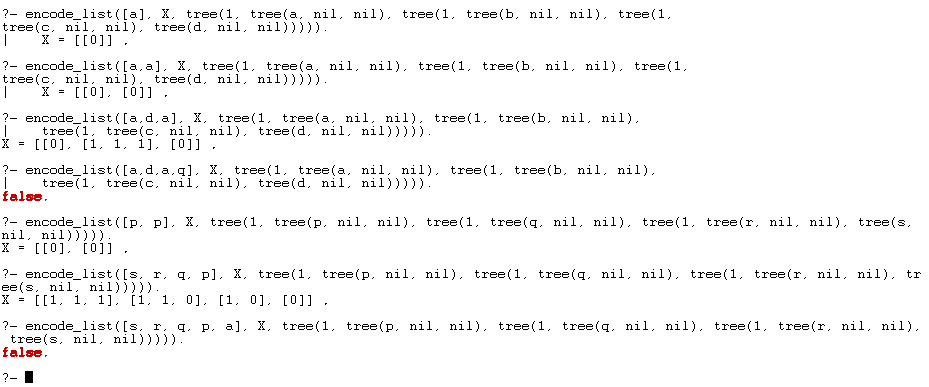
**8.0**



**8.1**



**8.2**



**8.3**

