Ejercicios para trabajar con listas

Ejercicio 01:

Escriba una función recursiva que permita calcula el factorial de un número. Esta función recibe un número como parámetro y retorna el resultado del factorial

factorial(n) -> int

El factorial de un número se puede calcular de forma recursiva de la siguiente forma:

n! = n * (n-1)!
0! = 1

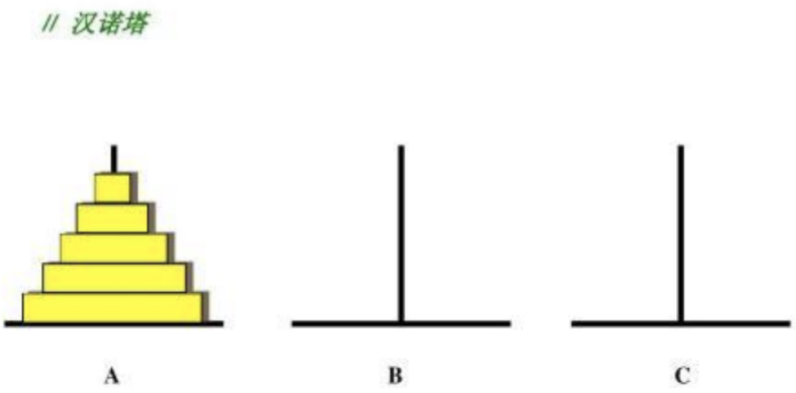
Escriba las líneas de código necesarias para pedir un número al usuario y llamar a la función factorial(n)

Ejercicio 02:

El problema de la Torre de Hanoi es un problema clásico. La Torre de Hanoi, también conocida como Torre de Hanoi, se originó a partir de una antigua leyenda en la India. Cuando Brahma creó el mundo, hizo tres pilares de diamantes. En un pilar, se apilaron 64 discos de oro en orden de tamaño de abajo hacia arriba. El Brahma ordenó al Brahmin que volviera a colocar el disco en otro pilar en orden de tamaño desde abajo. También estipula que en cualquier momento, el disco no se puede agrandar en el disco pequeño, y solo se puede mover un disco entre los tres pilares a la vez.
luego se convirtió en el juego Tower of Hanoi:

1. Hay tres pilares, hay n discos en el pilar A y el más grande está debajo del más pequeño.
2. Solo se puede mover un disco a la vez y el disco pequeño solo se puede colocar en el disco grande.
3. Mueva el disco del pilar A al pilar C.

Resuelva el número de movimientos para mover n discos al pilar C.



Ejercicio 03:

Diseñar un función recursiva que dado un vector de números enteros retorne la suma de sus elementos.

suma_lista(lista) → int

```
datos del problema [1, 20, 3, 4, 50]
suma de los elementos 78
```

Ejercicio 04:

Diseñar una función recursiva que escriba al revés la cadena de caracteres que se le pasa como parámetro.

invertir(frase) → string

```
Frase original: onomatopéyico
Frase original: ociyépotamono
```

Ejercicio 05:

Diseñar una función recursiva tal que, dados dos listas de números enteros, retorne un booleano indicando si son iguales, es decir, si tienen los mismos valores en las mismas posiciones.

comparar_listas(lista1, lista2) → booleano

Ejercicio 06:

Diseñar una función recursiva tal que, dada una lista de números enteros, retorne el mayor de los números que hay en ella.

encontrar_mayor(lista1) → int

Ejercicio 07:

Diseñar una función recursiva tal que, dada una lista de números enteros, retorne el menor de los números que hay en ella.

encontrar_menor(lista1) → int

Ejercicio 08:

Diseñe una función recursiva que permita resolver el siguiente problema:

El algoritmo chino de multiplicación. Diseñar un método que multiplique dos números enteros usando las siguientes equivalencias:

$$x * y = (2 * x) * \left(\frac{y}{2}\right) = \begin{cases} (2 * x) * (y \ div \ 2), & si \ y \ es \ par \\ (2 * x) * (y \ div \ 2) + x, & si \ y \ es \ impar \end{cases}$$

Ejercicio 09:

La sucesión de Fibonacci viene definida por la siguiente recurrencia:

$$f_{n+2} = f_n + f_{n+1}$$

con valores iniciales $f_0 = 0$ y $f_1 = 1$.

Diseñad e implementad un método recursivo para calcular el enésimo término de la sucesión y mostrad el árbol de llamadas que se produce al calcular f_4 con vuestra solución.

Ejercicio 10:

Implemente una función recursiva que permita transformar un número decimal en su correspondiente binario.

pasar_binario(decimal)

Ejercicio 11:

Sólo la función recursiva que verifique si un string es palíndrome:

Implemente, tanto de forma recursiva como de forma iterativa, una función que nos diga si una cadena de caracteres es simétrica (un palíndromo). Por ejemplo, "DABALEARROZALAZORRAELABAD" es un palíndromo.

verifica_palindrome(string) -> booleano