EJERCICIOS DE TÉCNICAS BÁSICAS PROGRAMACIÓN, funciones 2

Nota: es todos estos ejercicios la interfaz debe ser lo más sencilla posible, en el sentido de que lo que importa es que el *script* introduzca las entradas indicadas y las salidas, pero sin preocuparse de detalles de presentación.

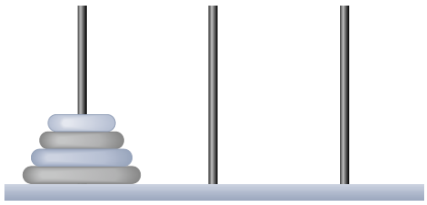
1. El **factorial** de un número natural N (1, 2, 3, 4, ...) se representa como N! y es igual a N * (N-1) * (N-2) * ... * 2 * 1. Por ejemplo: 5! = 5 · 4 · 3 · 2 · 1. También se considera que 0! es 1. Podemos observar que N! = N * (N-1)!.

Escribe 2 funciones factorial1 y factorial2 que calculen el factorial de un número N.

- factorial1: lo hará de **forma iterativa**
- factorial2: lo hará de **forma recursiva**

Prueba cada una de las funciones con los números siguientes: 5, 7, 10, 20, 100. Para cada invocación de función "cronometra" cuánto tiempo lleva cada versión de la función usando la Performance API (https://developer.mozilla.org/en-US/docs/Web/API/Performance/now)

2. **Problema de las Torres de Hanoi**.



Toda o todo aprendiz de programación pasa por este problema clásico: las Torres de Hanoi. Cuenta la leyenda que en un templo del Lejano Este, los monjes se encuentran con el problema de mover una pila de discos de un poste a otro poste. La pila inicial de discos tiene 64 discos apilados de mayor (en la base de la pila) a menor (en la cima de la pila) (ver por favor el gráfico). Las reglas que deben seguir los monjes son que deben mover los discos del primer poste al segundo poste bajo las condiciones siguientes:

- solo se puede mover 1 disco en cada desplazamiento de un poste a otro
- nunca se permite colocar un disco de mayor tamaño sobre otro más pequeño.
- existe un tercer poste que permite colocar discos temporalmente (pero siguiendo las mismas reglas).

Se dice que el mundo llegará a su final antes de que los monjes consigan terminar su tarea.

Supón que los monjes quieren mover los discos del poste 1 al poste 3. Nos solicitan desarrollar un algoritmo que imprima la secuencia precisa de movimientos de disco a disco de un poste a otro poste.

Si intentásemos resolver este problema con métodos convencionales de razonamiento, muy probablemente nos acabaríamos embrollando seguramente al poco rato. En su lugar, si buscamos una solución al problema usando **recursión** (es decir, pensando cómo descomponer un problema de tamaño N en otro/s iguales pero de tamaño más pequeño) en seguida aparece la solución.

La idea es la siguiente: mover N discos del poste 1 al poste 3, se puede ver como:

a) mover N-1 discos del poste 1 al poste 2, usando el poste 3 para almacenamiento temporal
b) mover el último disco (el más grande) del poste 1 al poste 3
c) mover los N-1 discos del poste 2 al poste 3, usando el poste 1 para almacenamiento temporal

, y este proceso termina cuando N sea igual a 1 (que es lo que se llama en recursión el **caso base**).

Escribe un programa para resolver el problema de las Torres de Hanoi, usando una función recursiva que tenga los siguientes 4 parámetros:

a) número de discos a mover
b) el poste en los que los discos están inicialmente engarzados
c) el poste al cual se deben mover todos los discos (respetando las normas previamente indicadas)
d) el poste usado para almacenamiento temporal.

Tu script debe imprimir las instrucciones precisas para mover los discos de un poste a otro: por ejemplo, para mover 3 discos del poste 1 al poste 3, tu programa debe imprimir la siguiente serie de movimientos:

1 --> 3
1 --> 2
3 --> 2
1 --> 3
2 --> 1
2 --> 3
1 --> 3

Prueba la solución del problema para 8, 16, 32 y 64 discos. Para cada invocación de función "cronometra" cuánto tiempo lleva el cómputo de la función usando la Performance API, pero desactivando el código que imprime los movimientos de

los discos (https://developer.mozilla.org/en-US/docs/Web/API/Performance/now)