GUÍA DE EJERCICIOS # 3 - RECURSIÓN

Resuelva todos los ejercicios de esta guía usando prioritariamente funciones recursivas.

Ejercicio 1
La forma para calcular cuantas maneras diferentes tengo para elegir r cosas distintas de un conjunto de n cosas es:

C(n,r) = n! (r!*(n-r)!)

Donde la función factorial se define como
n! = n *(n-1)*(n-2)*…*2*1

Descubra una versión recursiva de la fórmula anterior y escriba una función recursiva que calcule el valor de dicha fórmula.

Ejercicio 2
Escriba una función recursiva que ordene de menor a mayor una lista de enteros basándose en la siguiente idea: coloque el elemento más pequeño en la primera ubicación, y luego ordene el resto del arreglo con una llamada recursiva.

Ejercicio 3
Escribir una función recursiva que devuelva la suma de los primeros N enteros

Ejercicio 4
Escribir un programa que encuentre la suma de los enteros positivos pares desde N hasta 2. Chequear que si N es impar se imprima un mensaje de error.

Ejercicio 5
Escribir un programa que calcule el máximo común divisor (MCD) de dos enteros positivos. Si M >= N una función recursiva para MCD es
MCD = M si N =0
MCD = MCD (N, M mod N) si N <> 0
El programa le debe permitir al usuario ingresar los valores para M y N desde la consola. Una función recursiva es entonces llamada para calcular el MCD. El programa entonces imprime el valor para el MCD. Si el usuario ingresa un valor para M que es < que N el programa es responsable de *switchear* los valores.

Ejercicio 6
Programe una función recursiva que transforme un número entero positivo a notación binaria.

Ejercicio 7
Programe una función recursiva que transforme un número expresado en notación binaria a un número entero.

Ejercicio 8
Programe una función recursiva que calcule la suma de un arreglo de números enteros.

Ejercicio 9
Programe una función recursiva que invierta los números de un arreglo de enteros.

Ejercicio 10
Implemente una función recursiva que nos diga si una cadena es palíndrome.

Ejercicio 11
Diseñe e implemente un algoritmo que imprima todas las posibles descomposiciones de un número natural como suma de números menores que él.
1= 1
2 = 1+1
3= 2 + 1
3= 1+1+1
4= 3+1
4= 2+1+1
4 = 1+1+1+1
4=2+2
4=2+1+1
4=1+1+1+1
N = (n-1) +1
N = (n-2) + 2 = (n-2) + 1 + 1

Ejercicio 12
Escribir una función que reciba un número positivo n y devuelva la cantidad de dígitos que tiene.

Ejercicio 13
Escribir una función que simule el siguiente experimento:
Se tiene una rata en una jaula con 3 caminos, entre los cuales elige al azar (cada uno tiene la misma probabilidad), si elige el 1 luego de 3 minutos vuelve a la jaula, si elige el 2 luego de 5 minutos vuelve a la jaula, en el caso de elegir el 3 luego de 7 minutos sale de la jaula. La rata no aprende, siempre elige entre los 3 caminos con la misma probabilidad, pero quiere su libertad, por lo que recorrerá los caminos hasta salir de la jaula.
La función debe devolver el tiempo que tarda la rata en salir de la jaula.

Ejercicio 14
Escribir una función que reciba 2 enteros n y b y devuelva True si n es potencia de b.

Ejercicio 15
Escribir una funcion recursiva que reciba como parámetros dos strings a y b, y devuelva una lista con las posiciones en donde se encuentra b dentro de a.

Ejercicio 16
Escribir dos funciones mutualmente recursivas par(n) e impar(n) que determinen la paridad del numero natural dado, conociendo solo que:

- 1 es impar.
- Si un número es impar, su antecesor es par; y viceversa.

Ejercicio 17
Escribir una función que calcule recursivamente el n-ésimo número triangular (el número 1 + 2 + 3 + ... + n).

Ejercicio 18
Escribir una funcion recursiva que encuentre el mayor elemento de una lista.

Ejercicio 19
Escribir una función recursiva para replicar los elementos de una lista una cantidad n de veces. Por ejemplo, replicar ([1, 3, 3, 7], 2) = ([1, 1, 3, 3, 3, 3, 7, 7])

Ejercicio 20
Programar un algoritmo recursivo que permita hacer la división por restas sucesivas.

Ejercicio 21
Programar un algoritmo recursivo que permita invertir un número.

Ejercicio 22
Programar un algoritmo recursivo que permita sumar los dígitos de un número

Ejercicio 23
Programar un algoritmo recursivo que permita hacer una multiplicación, utilizando el método Ruso (investigar).

Ejercicio 24
Implementae una función recursiva que imprima por pantalla los valores desde un número introducido por el usuario hasta 1.

Ejercicio 25
Implementae una función recursiva que devuelva el resultado de la siguiente expresión, para un número natural x pasado por parámetro:

$$f(x) = x^2 + (x-1)^2 + (x-2)^2 + \ldots + 2^2 + 1^2$$

Ejercicio 26
Implementar una función recursiva que lea desde teclado hasta encontrar un carácter de salto de línea (\n), y muestre por pantalla lo que ha leído, pero al revés.

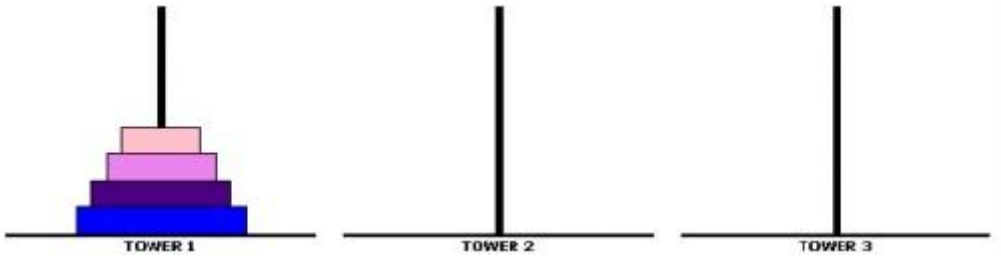
Ejercicio 27
Implementar una función recursiva que, dado un número natural, devuelva otro número que tenga los dígitos del primero, pero al revés. Ayuda: utiliza un parámetro auxiliar, donde vayas acumulando los resultados parciales.

Ejercicio 28
Cambio de moneda. Diseñe un algoritmo recursivo que dada una cierta cantidad M, efectúe el cambio a monedas de 2, 1, 0.5, 0.2, 0.1, 0.05, 0.02 y 0.01 euros, devolviendo el número de monedas de cada tipo que constituyen el cambio (eventualmente cero para alguno de los tipos). Se desea que el número de monedas obtenido sea el mínimo. Los únicos operadores disponibles son los de suma y resta.

Ejercicio 29
Implementa una función recursiva que devuelva true si el número que se le pasa como parámetro es primo y false en caso contrario.

Ejercicio 30
Simular el bucle FOR mediante procesos recursivos.

Ejercicio 31
Escenario: existen tres cilindros verticales donde es posible insertar discos. En uno de los cilindros hay n discos todos de diferente tamaño, colocados en orden de tamaño con el más chico arriba. Los otros dos cilindros están vacíos.



El problema es pasar la torre de discos completa a otro de los cilindros usando como único movimiento elemental el cambio de un disco de un cilindro a otro cualquiera, sin que en ningún momento un disco vaya a colocarse encima de otro más chico que él.

Escriban el pseudocódigo de los procedimientos que consideren necesarios para imprimir las secuencias de movimientos elementales que resuelvan el problema de las torres de Hanoi, de cualquier tamaño. Los movimientos elementales son de la forma: MOVER DISCO x DEL cilindroY AL cilindroZ.

Ejercicio 32
Escriban una función booleana recursiva llamada SonIguales que reciba dos listas como parámetros y devuelva TURE si son iguales (mismos elementos en el mismo orden), o FALSE en caso contrario.

Ejercicio 33
Escriban una función recursiva llamada ExisteElemento que verifique si un elemento x se encuentra en una lista L.

Ejercicio 34
Escriban una función recursiva llamada Ocurrencia que cuente la cantidad de ocurrencias de un elemento x en una lista L.
Escriban una función recursiva llamada Suma que retorne la suma de los elementos de una lista de enteros L.

Ejercicio 35
Escriba una función recursiva llamada Eliminar que elimine el elemento x de la lista L.

Ejercicio 36
Escriban una función recursiva llamada InsertarOrdenado que inserte en forma ordenada un elemento x en una lista ordenada L.

Ejercicio 37
Escriban una función recursiva llamada OrdenarLista que ordene una lista L.

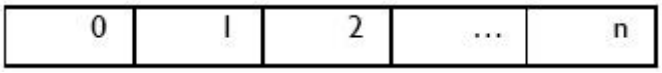
Ejercicio 38
Escriban una función recursiva llamada Merge que, a partir de dos listas ordenadas L1 y L2, genere una lista ordenada L3, a través de un proceso de intercalación de elementos (merge).

Ejercicio 39
Escriba una función recursiva llamada Invertir que, dada una lista L, la invierta.

Ejercicio 40
Escriba una función recursiva llamada Maximo que calcule el máximo de una lista de naturales L.

Ejercicio 41

En una hilera de una calle con adoquines unos niños juegan a la rayuela. Para esto numeran los adoquines de la siguiente forma:



Los movimientos permitidos del juego son:

- Al principio del juego los niños se ubican en el adoquín 0.
- De un adoquín numerado i se puede saltar al adoquín numerado i+1.
- De un adoquín numerado i se puede saltar al adoquín numerado i+2 (sin pasar por el adoquín i+1)
- Por ejemplo, el número de caminos posibles para n=3 es 3 y son los siguientes: (0,1,2,3), (0,2,3) y (0,1,3).

Escriban una función recursiva llamada CaminosPosibles que calcule el número de caminos posibles para alcanzar un adoquín objetivo numerado con n (mayor que cero).

Ejercicio 42
Escriban una función recursiva llamada MCD que calcule el Máximo Común Divisor de 2 naturales a y b (con a >= b).