RECURSIÓN EXPLÍCITA

Define las siguientes funciones de forma recursiva, sin usar funciones de orden superior (map, filter, fold, etc.).

1. Eliminar Duplicados

Devuelve una lista sin elementos duplicados, manteniendo el orden de la primera aparición.

eliminarDuplicados :: Eq a => [a] -> [a]

2. Aplanar Lista (Eficiente)

Aplane una lista de listas utilizando una función auxiliar con un acumulador para máxima eficiencia.

aplanarLista :: [[a]] -> [a]

3. Zip Con

Toma una función binaria y dos listas, aplicando la función a los elementos emparejados.

zipCon :: (a -> b -> c) -> [a] -> [b] -> [c]

4. Sublistas

Devuelve una lista de todas las sublistas contiguas (secuencias) de la lista de entrada.

sublistas :: [a] -> [[a]]

5. Take While

Devuelve el prefijo más largo de la lista cuyos elementos satisfacen el predicado.

takeWhile :: (a -> Bool) -> [a] -> [a]

6. Producto Lista

Calcule el producto de todos los números en una lista.

productoLista :: [Int] -> Int

7. Máximo Par por Suma

Tome una lista de pares [(Int, Int)] y devuelva el par donde la suma de sus componentes es máxima.

maximoPar :: [(Int, Int)] -> (Int, Int)

8. Es Prefijo

Devuelve True si la primera lista es un prefijo de la segunda.

esPrefijo :: Eq a => [a] -> [a] -> Bool

9. Drop While

Elimine el prefijo más largo de la lista que satisfaga el predicado.

dropWhile :: (a -> Bool) -> [a] -> [a]

10. Filtrar Índices

Filtre los elementos de una lista cuyos índices satisfacen un predicado dado (usa una función auxiliar con contador).

filtrarIndices :: (Int -> Bool) -> [a] -> [a]

LISTAS POR COMPRENSIÓN

Define las siguientes funciones utilizando exclusivamente la sintaxis de Listas por Comprensión.

1. Obtener Diagonal

Devuelve la lista de elementos en la diagonal principal de una matriz cuadrada.

obtenerDiagonal :: [[Int]] -> [Int]

2. Todos Perfectos

Dado un límite n, devuelva una lista de todos los números perfectos menores o iguales a n.

todosPerfectos :: Int -> [Int]

3. Primos Gemelos

Dado un límite n, devuelva una lista de todos los pares de primos gemelos menores o iguales a n.

primosGemelos :: Int -> [(Int, Int)]

4. Vector Suma

Devuelva la suma vectorial de dos listas de Int de igual longitud. (Pista: Usa la función 'zip').

vectorSuma :: [Int] -> [Int] -> [Int]

5. Posiciones De

Devuelva una lista de los índices (posiciones) donde se encuentra un elemento dado.

posicionesDe :: Eq a => a -> [a] -> [Int]

6. Permutaciones

Genere todas las listas que son permutaciones de elementos tomados de una lista de listas. (Ej: [[1, 2], [3, 4]] -> [[1, 3], [1, 4], [2, 3], [2, 4]]).

permutaciones :: [[a]] -> [[a]]

7. Aplanar Matriz

Devuelva una lista simple con todos los elementos de una matriz (lista de listas).

aplanarMatriz :: [[a]] -> [a]

8. Generar Tablero

Dado un tamaño n, genere las coordenadas (x, y) de un tablero de ajedrez de n x n.

generarTablero :: Int -> [(Int, Int)]

9. Filtrar Pares Suma

Devuelva solo las tuplas (Int, Int) cuya suma de componentes es un número par.

filtrarParesSuma :: [(Int, Int)] -> [(Int, Int)]

10. Divisores Primos

Devuelva la lista de todos los divisores de un número n que también son números primos.

divisoresPrimos :: Int -> [Int]

ALGORITMOS DE ORDENAMIENTO

Implementa los siguientes algoritmos y funciones auxiliares.

1. Insertion Sort

Implementa el algoritmo de ordenamiento por inserción.

insertionSort :: Ord a => [a] -> [a]

2. Is Sorted

Implementa la función que devuelve True si una lista está ordenada de manera ascendente (polimórfica y recursiva).

isSorted :: Ord a => [a] -> Bool

3. QuickSort Tupla

Implementa Quicksort para ordenar una lista de tuplas [(Int, a)] basándose únicamente en el primer elemento (Int).

quickSortTupla :: Ord a => [(Int, a)] -> [(Int, a)]

4. Merge Sort

Implementa el algoritmo Merge Sort completo.

mergeSort :: Ord a => [a] -> [a]

5. Partition By

Generaliza la función 'particionar' para que tome un predicado (a -> Bool) y divida la lista en dos.

partitionBy :: (a -> Bool) -> [a] -> ([a], [a])

6. Selection Sort

Implementa el algoritmo Selection Sort utilizando la función minimoYResto.

selectionSort :: [Int] -> [Int]

7. Delete (Auxiliar)

Implementa la función que elimina la primera ocurrencia de un elemento x de una lista xs.

delete :: Eq a => a -> [a] -> [a]

8. Is Permutation

Devuelve True si dos listas son permutaciones una de la otra. (Pista: usa el ordenamiento).

isPermutation :: Ord a => [a] -> [a] -> Bool

9. Mínimo General (Auxiliar)

Implementa la función que encuentra el mínimo en una lista [a] donde a puede ser cualquier tipo ordenable.

minimoGeneral :: Ord a => [a] -> a

10. Maximum By

Implementa una función que encuentre el elemento máximo en una lista usando una función de comparación externa (a -> Int).

maximumBy :: (a -> Int) -> [a] -> a