Para todos los siguientes ejercicios, realizar la solución utilizando recursividad en lugar de iteraciones:

1) rangoHasta(n) -> Lista de números: dado un número "n", retorna la lista de números desde el 0 hasta el N incluído. Por ejemplo: rangoHasta(5) -> [0,1,2,3,4,5].

2) rango(desde, hasta) -> lista de números: similar a rango, pero ahora se puede especificar el "desde". Ej: rango(5, 10) -> [5,6,7,8,9,10]. No hace falta validar que desde sea menor a hasta o tener rangos decrecientes.

3) sumaHasta(n) -> numero. Retorna la suma de los numeros desde el 0 hasta el N. Por ejemplo. sumaHasta(5) = 5 + 4 + 3 + 2 + 1 + 0 => 15

4) removerTodos(lista, elemento): -> lista, Dada una lista y un elemento, retorna otra lista igual a la original, pero sin el "elemento" dado. En caso en que el elemento aparezca múltiples veces, lo remueve de todas. Ejemplo:
remover([1,2,3,1,6,7,1,9,1], 1) -> [2,3,6,7,9]

5) aparear(unaLista, otra) -> lista de pares (x, y): tal que "x" pertence a "unaLista", e y pertenece a "otra". Ejemplo: aparear([1,2,3], ['a','b','c']) -> [(1,'a'), (2,'b'), (3,'c')]

6) aMayusculas(unString) -> otro string igual pero en mayusculas. Acordarse que los strings también se pueden tratar como listas

7) sumarN(n, numeros): Realizar una función que dada una lista de números y un número N, retorna la suma de todos los N primeros elementos. Ejemplo sumarN(3, [2, 4, 6, 8, 10, 12]) -> 2+4+6 = 24

8) ordenar(numeros): Realizar una función que dada una lista de números retorne otra lista con los mismos números pero ordenados de menor a mayor. Pista: en una lista ordenada siempre se da que un elemento en la posición x es el mínimo de la sublista que sigue. Ej: ordenar([3,6, -1]) -> [-1,3,6]

9) sinDuplicados(lista): Realizar una función recursiva que dada una lista, retorna la lista de esos elementos sin duplicados. Recordar que dado un elemento, sabemos que es duplicado si está en otra parte de la lista. Recuerden el operador "in". Ej: unNumero in numeros.

10) sinDuplicadosConSemilla(lista, resultado): Realizar otra función con el mismo objetivo que sinDuplicados, pero utilizando una semilla "resultado", de modo de evitar generar nuevas listas en cada paso y en su lugar ir acumulando el resultado en la semilla (logrando así la recursividad de cola).

11) mezclar(lista, otra) -> lista: Dadas dos listas ordenadas, retorna una nueva que tiene los elementos de ambas, también ordenados.

Ej: mezclar([2, 10, 99], [-1, 8, 200]) -> [-1,2,8.10,99,200]

12) esPar(n) -> boolean: Dado un numero dice si ese número es par o no.

Para esto recordar que el caso más básico es el 0 que es par. Luego un número es par si su anterior es impar. Y un número es impar si su anterior es par.
Ej esPar(4) si esImpar(3)
esImpar(3) si esPar(2)
esPar(2) si esImpar(1)
esImpar(1) si esPar(0)
esPar(0) ? SI !