Ejercicio 1.- Extender la especificación de listas LISTA2[ELEMENTO], vista en clase, añadiendo las siguientes operaciones (pueden ser parciales):

- eliminar: elemento lista → lista, que elimina todas las apariciones de un elemento en una lista.
- repeticiones: elemento lista → natural, para calcular el número de veces que aparece un elemento en una lista.
- _==_: lista lista → bool, que determina si dos listas son iguales.

a) Escribir en pseudocódigo estas operaciones partiendo únicamente de las operaciones de la especificación vistas en clase.
b) Escribir en pseudocódigo estas operaciones utilizando la representación con memoria dinámica vista en clase.

Ejercicio 2.- Escribir en pseudocódigo la operación insertar_en_orden, que inserta un elemento en una lista ordenada (Ejemplo 5 en Tema 4 Listas Básicas).

a) Utilizando la implementación con memoria dinámica para lista simplemente enlazada vista en clase.
b) Utilizando la implementación con memoria dinámica para lista doblemente enlazada vista en clase.

Ejercicio 3.- Dar la especificación del TAD básico LISTA[BOOLEAN]. Extender dicha especificación con operaciones adicionales para:

c) maximo_seguidos: lista → natural, que obtiene cuál es la mayor cantidad de booleanos iguales seguidos que se encuentra en la lista; por ejemplo (simplificado), maximo_seguidos(FFTFTTFFFTTF) = 3 por la secuencia FFF.
d) reducir_datos: lista → lista, que reduce todas las secuencias de booleanos iguales que están seguidos a un único dato, es decir, se reducen los trozos seguidos; por ejemplo (simplificado), reducir_datos(FFTFTTFFFTTF) = FTFTFTF.

Ejercicio 4.- Suponiendo conocidas las operaciones de los ejercicios 2 y 3, incluyendo la operación _≤_: elemento elemento→ bool, especificar operaciones para

- ordenar una lista de menor a mayor usando el método de selección.
- ordenar una lista de menor a mayor usando el método de inserción (aunque no es necesario, puede ser útil tener un acumulador para las ordenaciones parciales).
- ordenar una lista de menor a mayor usando el método de ordenación rápida o Quicksort, separando los datos de la lista en “pequeños” (menores que un pivote) y “grandes” (mayores que un pivote).

Ejercicio 5.- Suponiendo que los elementos de las listas pueden compararse con una operación _≤_: elemento elemento → bool, que comprueba si un elemento es menor o igual que otro, extender la especificación de listas LISTA2[ELEMENTO], vista en clase con las operaciones (pueden ser parciales):

- mínimo: lista → elemento, que busca el menor elemento de una lista.
- ordenada?: lista → bool, que comprueba si una lista está ordenada.
- quitar: elemento lista → lista, que quita una aparición cualquiera de un elemento que esté en la lista:
  - suponiendo que la lista no está ordenada.
  - suponiendo que la lista está ordenada.

Ejercicio 6.- Extender la especificación del TAD LISTA[ELEMENTO] creando las dos siguientes operaciones (pueden ser parciales):

- obtener la lista formada por los n primeros elementos de una lista
- obtener la lista resultante de quitar los n primeros elementos de una lista.

a) Escribir en pseudocódigo estas operaciones partiendo únicamente de las operaciones de la especificación vistas en clase.
b) Escribir en pseudocódigo estas operaciones utilizando la representación con memoria dinámica vista en clase.

Ejercicio 7.- Escribir en pseudocódigo, utilizando la representación con memoria dinámica para listas doblemente enlazadas vista en clase, las operaciones del TAD LISTA+[ELEMENTO] (listas ampliadas):

- parcial _ [_]: lista natural → elemento, devuelve el elemento que ocupa la posición dada de la lista.
- parcial insertar: elemento lista natural → lista, inserta un elemento en la lista en la posición dada.
- parcial modificar: elemento lista natural → lista, modifica la lista sustituyendo el elemento de la posición dada.
- parcial borrar: lista natural → lista, borra el elemento que ocupa la posición dada de la lista.e
- está?: elemento lista → bool buscar: elemento lista → natural, comprobar si un dato está en la lista, y en qué posición se encuentra.

Ejercicio 8.- Suponiendo conocida la especificación completa de LISTAS[ELEMENTO] (usando como generadoras [] y _:_), y que en el TAD de *elemento* se tiene una operación _==_: elemento elemento → bool, se pide especificar las operaciones necesarias para:

- contar cuántos elementos distintos hay en la lista.
- recolocar los elementos de la lista para que todos los elementos iguales entre sí sean consecutivos.
- poder determinar si dos listas de igual longitud tienen los mismos elementos (no necesariamente en el mismo orden).

a) Escribir en pseudocódigo estas operaciones partiendo únicamente de las operaciones de la especificación vistas en clase.
b) Escribir en pseudocódigo estas operaciones utilizando la representación con memoria dinámica vista en clase.