Estrategias de Programación y Estructuras de Datos
Tema 4: Listas
Ejercicios propuestos

1. Calcular el coste de todas las operaciones públicas (no abstractas) de `Collection<E>`.

2. Calcular el coste de todas las operaciones públicas de `Sequence<E>`.

3. Calcular el coste de todas las operaciones públicas de `List<E>`.

4. Realizar una implementación de `ListIPIF<E>` (Listas con Punto de Interés) utilizando dos pilas como Estructura de Datos de soporte. Calcular el coste de todas las operaciones públicas de esta implementación.

5. Programar dos versiones de un método:

   ```
   ListIF<E> invierte(ListIF<E> l)
   ```

   que invierta la lista dada por parámetro.

   a) Primera versión: utilizando iteradores.
   b) Segunda versión: sin utilizar iteradores.

   Compare el coste asintótico temporal en el caso peor de ambas implementaciones.

6. Programar un método que determine si una lista está ordenada o no.

7. Programar un método que determine si en una lista existe un rellano (definido como un cierto número de elementos contiguos que son iguales), bajo los siguientes supuestos:

   a) Sabemos que la lista está ordenada.
   b) No sabemos si la lista está ordenada.

8. Programar un método:

   ```
   ListIF<E> mezclar(ListIF<E> a, ListIF<E> b)
   ```

   que recibe dos listas ordenadas crecientemente y devuelve el resultado de mezclar ambas listas manteniendo el orden.

9. Implementar una clase `SequenceMS<E>` que implemente el interfaz `SequenceMSIF<E>` (ver ejercicio 3 del tema 1), de secuencias con tamaño máximo limitado, utilizando un Array como estructura de datos de soporte. Calcule el coste asintótico temporal en el caso peor de todos sus métodos públicos y compárelo con el coste de los métodos equivalentes de `Sequence<E>`.

10. Implementar una clase `ListMS<E>` que extienda la clase `SequenceMS<E>` y que implemente el interfaz `ListMSIF<E>` (ver ejercicio 4 del tema 1). Compare el coste asintótico temporal en el caso peor de todos sus métodos públicos con los correspondientes de `List<E>`.