EXAMEN ENERO 2025

SE VALORARÁ POSITIVAMENTE:
1. Los algoritmos y las estructuras de datos utilizados para la solución así como su justificación.
SE PENALIZARÁ:
2. Los errores en los requisitos y los algoritmos, teniendo en cuenta que un error leve descuenta 1 punto y un error grave 2 puntos. • Se considera un error leve aquel, que pese a su existencia, permite una solución al problema. • Un error grave es aquel que imposibilita una correcta y exacta solución al problema. • La no utilización del método, de cada ejercicio, en un programa principal será un leve. • En cuánto a la pregunta teórica su valor es de 1/2 punto. Valiendo el resto del ejercicio 1 punto y medio. En este caso grave = - 1,5.
Todos los ejercicios valen 2 puntos y se debe respetar el tiempo máximo establecido y las normas de permanencia. Recomendable utilizar, como método de resolución, el visto en clase pero se corregirá sobre Java.

Nombre y apellidos: **30/01/2025** **Firma:**

1. Implementad el método de la **búsqueda binaria recursiva** para encontrar la posición (índice) de un elemento en un array ordenado de manera ascendente de números enteros. Está prohibido el uso de su forma iterativa, eso quiere decir que no podemos usar bucles (for, while, do-while). Únicamente podemos utilizar condicionales. Considera que el método recibe un array de enteros, un entero que determina el límite por la izquierda, un entero que determina el límite por la derecha y finalmente el valor a buscar. Además, ten el que si el elemento no se encuentra, el método debe devolver un -1.

Ejemplos de uso:

- busquedaBinariaRecursiva({2, 4, 7, 10, 13}, 0 , 4 , 10) → 3
- busquedaBinariaRecursiva({2, 16, 23, 38}, 0 , 5 , 2) → 0
- busquedaBinariaRecursiva({2, 5, 8, 12, 16, 23, 38}, 0 , 6 , 50) → -1

2. Teniendo en cuenta esta estructura de datos de java:

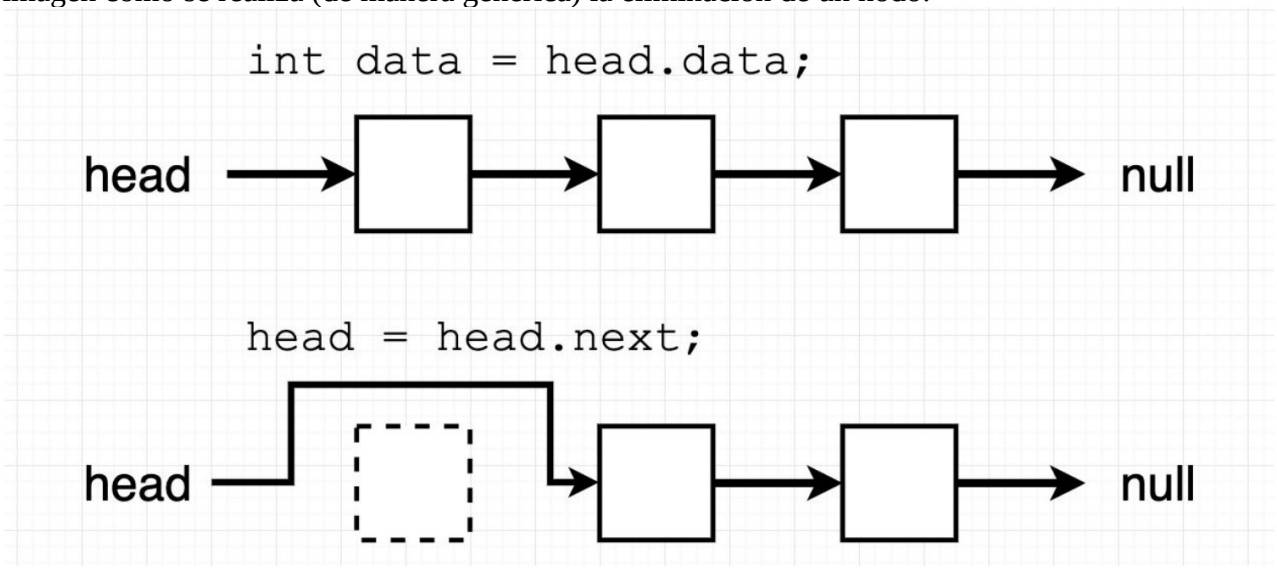
```
class Node {

int data;
Node next = null;

Node(final int data) {
  this.data = data;
 }

 }
```

Implementa un método que recibe un Nodo y que dada una lista enlazada de números, elimine todos los nodos que contengan un número par. No importa que implementes el método que imprime la lista resultante por pantalla. Solamente nos interesa el método eliminarPares. Podemos ver en esta imagen como se realiza (de manera genérica) la eliminación de un nodo:



Ejemplos de uso:

- Con la siguiente definición (cabecera) del método: *result = eliminarPares(head)*; si tenemos la lista con nombre result: **1** -> **2** -> **3** -> **4** -> **5** -> **null** , deberíamos eliminar los nodos con los valores 2 y 4, resultando en la lista llamada result: **1** -> **3** -> **5**-> **null**.

- Con la siguiente definición (cabecera) del método: *result = eliminarPares(head)*; si tenemos la lista con nombre result: **2** -> **3** -> **4** -> **6** -> **null** , deberíamos eliminar los nodos con los valores 2, 4 y 6 resultando en la lista llamada result: **3** -> **null**.

3. En matemáticas, una matriz Toeplitz es aquella en la que los elementos en cualquier diagonal dada desde la parte superior izquierda a la parte inferior derecha son **idénticos**.

Por ejemplo:

{{ 1, 2, 3, 4, 8 },
{ 5, 1, 2, 3, 4 },
{ 4, 5, 1, 2, 3 },
{ 7, 4, 5, 1, 2}}

Escribe un método llamado *isToeplitzMatrix* que recibe una matriz (o array bidimensional) y devuelve true dicha matriz es una matriz Toeplitz o false en caso contrario.

Ejemplos de uso:

*isToeplitzMatrix([[**1, 2, 3, 4, 8], [5, 1, 2, 3, 4], [4, 5, 1, 2, 3], [7, 4, 5, 1, 2]]**) → true*

*isToeplitzMatrix([[**9, 2, 3, 4, 8], [5, 7, 2, 3, 4], [4, 5, 1, 2, 3], [7, 4, 5, 1, 2]]**) → false*

4. Implementa un método de **ordenación** para un array bidimensional (o matriz) de números enteros. Este método debe recibir una array bidimensional como parámetro y permitir la ordenación tanto ascendente como descendente. El tipo de ordenación (ascendente o descendente) se especificará mediante un parámetro booleano. La elección del algoritmo de ordenación es **libre** y debes contestar a esta pregunta: ¿qué complejidad espacial tiene el método a implementado?

Ejemplos de uso:

- Dada esta matriz:

  [[9, 8, 5],
  [4, 2, 1]]

  Al ordenarla **descendentemente** quedaría así:

  [[1, 2, 4],
  [5, 8, 9]]

- Dada esta matriz:

  [[10, 8, 5]
  [4, 2, 1]]

  Al ordenarla **ascendentemente** quedaría así:

  [[10, 8, 5]
  [4, 2, 1]]

5. Un equipo botánico que ha recopilado una lista de nombres de especies de plantas observadas durante una expedición te ha pedido un desarrollo para solucionar un problema. Debido a errores en la transcripción, algunas especies se han registrado más de una vez, y desean identificar cuántas especies distintas tienen entradas duplicadas.
Escribe un método llamado **especiesDuplicadas** que recibe un array de cadenas, donde cada cadena representa el nombre de una especie de planta. El método debe devolver el número de especies únicas que aparecen más de una vez en el array. Los nombres de las especies no son key sensitive, es decir, no distinguen entre mayúsculas y minúsculas, por ejemplo, "rosa" y "Rosa" deben considerarse la misma especie.

Ejemplos de uso:

- especiesDuplicadas(["Rosa", "Lirio", "rosa", "Tulipan", "LIRIO", "Margarita"]) → 2
- especiesDuplicadas(["Orquidea", "Girasol", "Clavel", "Azucena"]) → 0
- especiesDuplicadas(["Bambu", "bambu", "BAMBU", "Bambu"]) → 1