# Grado en Ingeniería Informática y Grado en Estadística
# Fundamentos de Programación

**Examen convocatoria extraordinaria. 7 de febrero de 2022.**

**Apellidos** _____

**Nombre** _____ **Grupo: T2**

**DNI y Firma**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

| | |
|---|---|
| Duración del examen: 3 horas. | Se valorará la presentación y la claridad en la exposición. |
| Empezar cada problema en una cara distinta. | Se valorará la calidad de las soluciones. |
| Poner nombre y apellidos en todas las páginas. | No se calificarán las respuestas escritas a lápiz. |
| Las hojas del enunciado también deben entregarse. | No se deben entregar las hojas que se usen como borrador. |

**1.** **[8 ptos]** En un determinado juego de bingo se utilizan cartones de 4 filas y 9 columnas en el que en cada columna aparecen solo 2 valores, en orden creciente, en posiciones aleatorias, que se corresponden con la misma decena (en la columna 1 solo hay valores del 1 al 10, en la 2 valores del 11 al 20, en la 3 valores del 21 al 30 y así hasta la última columna en la que solo hay valores del 81 al 90.

**a)** **[3 ptos]** Elaborar un método Java que construya y devuelva una matriz de enteros que represente un cartón de bingo con las restricciones especificadas. Las celdas vacías se representarán con un 0.
Un ejemplo de un cartón de bingo, representado por una matriz de enteros, podría ser:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | **3** | **0** | **21** | **0** | **42** | **0** | **63** | **78** | **85** |
| 1 | **0** | **12** | **0** | **33** | **0** | **0** | **69** | **0** | **0** |
| 2 | **7** | **0** | **0** | **0** | **0** | **54** | **0** | **80** | **0** |
| 3 | **0** | **19** | **23** | **37** | **48** | **60** | **0** | **0** | **90** |

Para representar un cartón de este tipo que permita ir tachando los números que salen en el juego, se puede usar una matriz de registros de tipo **`regElem`**, que contienen tres campos: uno de tipo **`boolean`** que indica si esa celda está o no vacía, otro **`int`** que guarda el número correspondiente a esa celda y otro **`boolean`** que indicará si ese número ha sido "tachado" o no por el jugador.

Por ejemplo, el contenido inicial de una matriz de este tipo que almacene los mismos datos (mismos números en las mismas celdas no vacías) que la matriz de enteros del apartado anterior, sería:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | No Vacío<br>**3**<br>No Tachado | Vacío<br>*Irrelevante*<br>*Irrelevante* | No Vacío<br>**21**<br>No Tachado | Vacío<br>*Irrelevante*<br>*Irrelevante* | No Vacío<br>**42**<br>No Tachado | Vacío<br>*Irrelevante*<br>*Irrelevante* | No Vacío<br>**63**<br>No Tachado | No Vacío<br>**78**<br>No Tachado | No Vacío<br>**85**<br>No Tachado |
| 1 | Vacío<br>*Irrelevante*<br>*Irrelevante* | No Vacío<br>**12**<br>No tachado | Vacío<br>*Irrelevante*<br>*Irrelevante* | No Vacío<br>**33**<br>No Tachado | Vacío<br>*Irrelevante*<br>*Irrelevante* | Vacío<br>*Irrelevante*<br>*Irrelevante* | No Vacío<br>**69**<br>No Tachado | Vacío<br>*Irrelevante*<br>*Irrelevante* | Vacío<br>*Irrelevante*<br>*Irrelevante* |
| 2 | No Vacío<br>**7**<br>No Tachado | Vacío<br>*Irrelevante*<br>*Irrelevante* | Vacío<br>*Irrelevante*<br>*Irrelevante* | Vacío<br>*Irrelevante*<br>*Irrelevante* | Vacío<br>*Irrelevante*<br>*Irrelevante* | No Vacío<br>**54**<br>No Tachado | Vacío<br>*Irrelevante*<br>*Irrelevante* | No Vacío<br>**80**<br>No Tachado | Vacío<br>*Irrelevante*<br>*Irrelevante* |
| 3 | Vacío<br>*Irrelevante*<br>*Irrelevante* | No Vacío<br>**19**<br>No Tachado | No Vacío<br>**23**<br>No Tachado | No Vacío<br>**37**<br>No Tachado | No Vacío<br>**48**<br>No Tachado | No Vacío<br>**60**<br>No Tachado | Vacío<br>*Irrelevante*<br>*Irrelevante* | Vacío<br>*Irrelevante*<br>*Irrelevante* | No Vacío<br>**90**<br>No Tachado |

Dadas estas condiciones, se pide:

**b)** **[0,5 ptos]** Definir el tipo de datos **`regElem`**.

**c)** **[0,75 ptos]** Elaborar un método Java que construya y devuelva una matriz de **`regElem`** a partir de una matriz de **`int`** como la del apartado **a)**.

**d)** **[3 ptos]** Elaborar un método Java que, a partir de una matriz de tipo **`regElem`**, repita la petición de un valor entre **0** y **90** por teclado. La entrada **0** indica que otro jugador *ha cantado bingo* y el juego ha terminado. Las entradas entre **1** y **90** representan los valores que han salido en el bombo, el método debe actualizar la matriz, tachando el valor si estuviese en ella. Supóngase que los datos de entrada son correctos y que no se introducen números repetidos.

La petición de datos por teclado debe continuar hasta que se tachen todos los elementos de la matriz (*cantar bingo*) u otro jugador *cante bingo* (se indicará con un **0** en la entrada de datos).

Cuando se tachen todos los elementos de una fila (*cantar línea*) el método lo indicará escribiendo un mensaje en pantalla. **Solo se puede cantar línea una vez**, pero después de completar una fila el método debe continuar hasta que el juego termine.

**e)** **[0,75 ptos]** Elaborar un método Java que, a partir de una matriz de tipo **`regElem`**, escriba en el fichero de texto **`datos.txt`** (en la misma línea, separados por un espacio en blanco) todos los números que han sido tachados en un determinado momento.

**RECORDATORIO:** Para declarar y abrir en modo escritura un fichero de texto:

```
PrintWriter <id_fich> = new PrintWriter (new FileWriter (<nombre_fich>));
```

Al abrir el fichero se puede producir la excepción **`IOException.`**

**2.** **[1 pto]** Una cadena de caracteres es *n-inventada* cuando está formada exclusivamente por dígitos y, además, los caracteres en posiciones simétricas respecto al centro de la cadena suman el valor *n*.

Por ejemplo, la cadena **`"1807"`** es *8-inventada*, ya que sus pares simétricos suman 8 (1+7 = 8 y 8+0 = 8), pero la cadena **`"5133"`** no es *8-inventada*. La cadena **`"6138572"`** también es *8-inventada*.

La **cadena vacía** no es *n-inventada* para ningún n.

Escriba y documente una función recursiva que, a partir de un dígito **`n`** y una cadena dada, determine si esa cadena es o no *n-inventada*.

**RECORDATORIO**: La documentación de la función **`substring`**, de la clase **`String`**, es la siguiente:

**`public String substring(int beginIndex, int endIndex)`**

Returns a new string that is a substring of this string. The substring begins at the specified beginIndex and extends to the character at index endIndex - 1. Thus the length of the substring is endIndex-beginIndex.

Examples: "hamburger".substring(4, 8) returns "urge". "smiles".substring(1, 5) returns "mile"

**Parameters:**

beginIndex - the beginning index, inclusive.
endIndex - the ending index, exclusive.

**Returns:**

the specified substring.

**Throws:**

**`IndexOutOfBoundsException`** - if the beginIndex is negative, or endIndex is larger than the length of this String object, or beginIndex is larger than endIndex.

**3.** **[1 pto]** Crear un método Java que, dada **`list`** (de tipo **`Nodo`**), una referencia a una **lista dinámica** que contiene números enteros en su campo **`dato`**, devuelva la media aritmética de los números múltiplos de 3 contenidos en dicha estructura.

Supóngase definida la clase **`Nodo`** como:

```
public class Nodo {
       int dato;
       Nodo sgte;
       // constructores, etc.
}
```