RETOS HACK A TON — CONFIDENCIAL

Primer reto.

Dada una matriz n x n, formada por distintos números positivos desde el 1 al n^2, se dirá que esta forma un cuadrado mágico si la suma de las filas, columnas y diagonales es igual a la constante mágica. La constante mágica de un cuadrado mágico depende solo de n y se define por la siguiente fórmula: **M = n*(n^2+1)/2**

Por ejemplo, la constante del cuadrado mágico de tamaño 3 es la siguiente.

```
Cuadrado mágico de tamaño 3.
----------------------------
        8   1   6
        3   5   7
        4   9   2

La suma de cada fila y cada columna y diagonal = 3*(3^2+1)/2 = 15
```

Objetivo: Dada una matrix N x N, reordena un array de números enteros para que estos formen un cuadrado mágico. Ten cuidado, la conversión de cualquier número A en B tendrá un coste |A-B|. Trata de lograr la conversión con el mínimo coste posible.

Ejemplo:

```
  Input          Output
---------       ---------
6   3   2       8   1   6
1   7   9       3   5   7
5   4   8       4   9   2
```

Lenguajes aceptados: C/C++, Python & Java.

Duración: 45 min.

Segundo Reto.

Dado un texto de libre elección del alfabeto español, genera el mismo input en ASCII-ART. Considera mínimo 10 letras del alfabeto y el espacio.

Ejemplo:

```
Letras escogidas: {h, z, b, k, w, o, n, c, t, a}
           Input: hack a ton


Output:
     #              #              #
     ###  ##   ###  # #     ##     ###  ###  ##
     # #  # #  #    ##      # #     #   # #  # #
     # #  ###  ###  #  #    ###     ##  ###  # #
```

Restricciones: **Queda prohibido definir explícitamente cada uno de los caracteres de manera individual.**

Lenguajes aceptados:C/C++, Python & Java.

Duración: 1h.

Tercer Reto.

Más información sobre Expresiones Regulares.

La empresa para la que trabajas empezó a desarrollar un parser de expresiones regulares hace tiempo, pero el programador principal y el jefe del proyecto, tuvieron un importante enfrentamiento por la manera que el programador tenía de hacerlo. Los dos fueron incapaces de resolver sus diferencias y el programador fue despedido. Enfadado de que le despidieran injustamente, empezó a insultar al jefe y borró prácticamente la mayoría del programa de manera irreversible, lo que complicó la situación mucho más.

Tu jefe, en cambio te pide que hagas el parser desde el principio. Le disgusta tanto la idea de pensar en el programador que despidió, que te prohibe hacerlo empleando cualquier estructura de datos clásica simplemente por orgullo personal.

El programa deberá recibir por entrada de teclado una cadena de caracteres y una expresión regular y comprobar si la cadena *completa* cuadra con la expresión. El parser deberá tener en cuenta:

```
Letras mayúsculas y minúsculas: "a-z", "A-Z"
Números:                        "0" a "9"
Símbolos de delimitación:       "[]","()", "-"
Caracteres comodín:             "."
Alternación:                    "|"
Repetición:                     "+", "*"
Indeterminación:                "?"
```

Ejemplos:

```
Expresión: [a-z]+?
    Input: aaaabbbb
   Output: "Correcto"
---------------------------
Expresión: [a-z][0-9]+[A-Z]
    Input: ss732A
   Output: "Incorrecto"
```

-**Restricciones**: Queda prohibido emplear **cualquier estructura de datos clásica para almacenar los símbolos del input.** Se considerará "estructura de datos clásica" cualquiera de la siguiente lista: **array, linked-list, stack, queue, tree, graph, table, set; o variantes que imiten explícitamente el funcionamiento de estas**. Independientemente del método que se emplee para analizar el input, **el backtracking está prohibido**.

Lenguajes aceptados: C/C++, Python & Java.
Duración: 1h 30m.
