# Universidad Nacional de Asunción - Facultad Politécnica

**Primer Examen Parcial de Algoritmo**
**Fecha: 05 de noviembre de 2024**
**Directivas y Recomendaciones**

- Los comentarios aclaratorios pueden ayudar en la corrección del examen, influyendo favorablemente en su calificación.
- La duración del examen es de 120 minutos.
- El código fuente para cada tema debe subirse al aula virtual de la materia (en EDUCA), en la actividad correspondiente en el VPL.

## Ejercicio 1: Ordenamiento de elementos de una matriz (50%)

Escriba un programa en Python que permita cargar por teclado dos valores `m`, `n` enteros y positivos (no hace falta validar). Luego, se debe considerar una matriz de `m` filas y `n` columnas, cuyos elementos serán generados de manera aleatoria (en el intervalo `[0,100]`).

El programa debe contener una función llamada `ordenarMatriz()`, que reciba la matriz y luego la procese para ordenar sus valores según la regla que se muestra en el siguiente ejemplo:

`Entrada:`

| | | | | |
|---|---|---|---|---|
| 34 | 23 | 63 | 27 | 72 |
| 56 | 8 | 33 | 42 | 11 |
| 78 | 21 | 86 | 6 | 29 |
| 22 | 75 | 10 | 30 | 13 |
| 54 | 77 | 36 | 74 | 55 |
| 28 | 22 | 56 | 41 | 1 |

`Salida:`

| | | | | |
|---|---|---|---|---|
| 1 | 28 | 29 | 63 | 72 |
| 6 | 27 | 30 | 56 | 74 |
| 8 | 23 | 33 | 56 | 75 |
| 10 | 22 | 34 | 55 | 77 |
| 11 | 22 | 36 | 54 | 78 |
| 13 | 21 | 41 | 42 | 86 |

El programa debe mostrar la matriz antes y después del ordenamiento.

## El criterio de corrección será el siguiente (sobre 100%):

- Lectura de las dimensiones de la matriz y carga de elementos (generados aleatoriamente) en ella: 15%
- Impresión de la matriz: 10%
- Implementación de la función `ordenarMatriz()` y su llamada desde la parte principal del programa: 70%
- Uso de comentarios: 5%
- Si no se crea la función `ordenarMatriz()`, sino que su funcionalidad se implementa en la parte principal del programa, se penalizará (-20%)
- Si el programa tiene errores de sintaxis, se analiza el proceso; y por cada error de sintaxis se penalizará (-5%).

## Ejercicio 2: Seguridad de contraseñas (50%)

Usted es contratado para trabajar en el departamento de seguridad de una empresa tecnológica. La primera tarea que le asignan consiste en determinar el nivel de seguridad de las contraseñas de los usuarios de la plataforma de la empresa.

Para ello, debe implementar una función en Python llamada `esDeAltaSeguridad()`, que reciba una cadena (que representa una contraseña) y determine si es de "*alta seguridad*" (retornando un valor lógico/booleano). Se considera que una contraseña es de alta seguridad si cumple con cada uno de los siguientes requisitos:

- Tiene al menos 12 caracteres
- Cuenta con al menos tres letras minúsculas (`a-z`)
- Contiene al menos tres letras mayúsculas (`A-Z`)
- Tiene al menos dos números (`0-9`)
- Incluye al menos uno de los siguientes símbolos: `*$%-?!#`

Ejemplos de contraseñas con alta seguridad: `HolaMunDo123$`, `Algo2024!PArc2`

Ejemplos de contraseñas sin alta seguridad: `abcABC123$`, `Hola*Mundo123`, `Hola*MunDo$-a`, `Algo2024PArc2`

De acuerdo con lo retornado por la función implementada, se debe mostrar un mensaje en pantalla indicando si la cadena de entrada (que se ingresa por teclado) es o no de alta seguridad.

## El criterio de corrección será el siguiente (sobre 100%):

- Obtener cantidad de letras minúsculas: 10%
- Obtener cantidad de letras mayúsculas: 10%
- Obtener cantidad de números: 10%
- Obtener cantidad de símbolos: 15%
- Implementación de la función `esDeAltaSeguridad()`: 40%
- Lectura de la cadena de entrada, uso de la función e impresión del mensaje de salida: 10%
- Uso de comentarios: 5%
- Si no se crea la función `esDeAltaSeguridad()`, sino que su funcionalidad se implementa en la parte principal del programa, se penalizará (-20%)
- Si el programa tiene errores de sintaxis, se analiza el proceso; y por cada error de sintaxis se penalizará (-5%).