UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Introducción a Ciencia de la Computación

Ejercicios para Práctica Calificada 2

Las siguientes preguntas se recogen de prácticas pasadas y están acompañadas de un indicador de dificultad que va desde el nivel 1 al 5.

Funciones

1. (Nivel 1) Deberás crear una función que se llame *"maravilla"*. Esta función recibirá 3 parámetros, y tendrá las siguientes características:

   - El primer parámetro sera un carácter, el cual puede ser alguno de estos símbolos:
     "+" (más)
     "-" (menos)
     "*" (por)
     "/" (entre)
   - El segundo y el tecer parámetros serán dos números enteros .
   - La función devolverá el resultado de la operación enviada.
   - Si el primer parámetro fuera "/" ("entre") y el segundo número un cero, deberá imprimir un mensaje de error y no devolverá nada.

2. (Nivel 1) Implemente una función que reciba un número como parámetro y haga las sumatorias de la siguiente forma:

   - Si el número es par la función debe calcular y retornar la suma de todos los números pares desde el 0 hasta el número pasado como parámetro.
   - Si el número es impar entonces la función debe calcular y retornar la suma de todos los números impares desde el 0 hasta el número pasado como parámetro.
   - Si el número es 0 o no se le pasa ningun párametro, debe retornar el valor de 0.

   Luego implemente un programa que utilize la función anterior, solicitando al usuario que ingrese un número e imprimiendo el resultado.

   Algunos ejemplos de diálogo de este programa serían:

```
Ingrese un número: 5
Resultado: 9
```

```
Ingrese un número: 6
Resultado: 12
```

```
Ingrese un número: 9
Resultado: 25
```

3. (Nivel 1) Dada una lista de números construya una función que extraiga una secuencia de números dada dos posiciones. Presente el resultado como la suma de esos números extraídos. Los números ingresados se incorporan como una lista en una sola línea.

   Algunos ejemplos de diálogo de este programa serían:

```
Input: 3 4 5 6 7 9 1 2 3 4
pos1: 3
pos2: 5

Output: 18
```

```
Input: 3 4 1 2 7 9 1 2 3 4
pos1: 5
pos2: 8

Output: 19
```

4. (Nivel 1) Implemente una función que reciba un número entero como parámetro y retorne el factorial de dicho número. En caso no se envíe ningún valor al momento de invocarla, deberá calcular el factorial de 10. Algunos ejemplos de diálogo de este programa serían:

```
Input: 3
Output: 6
```

```
Input: 5
Output: 120
```

```
Input:
Output: 3628800
```

5. (Nivel 2) Implemente una función que reciba un número entero como parámetro y retorne la sumatoria de dicho número. En caso no se envíe ningún valor al momento de invocarla, deberá calcular la sumatoria de 10.

   Algunos ejemplos de diálogo de este programa serían:

```
Input: 5
Output: 15
```

```
Input: 10
Output: 55
```

6. (Nivel 2) Implementar un algoritmo que recibe un número entero menor a 9 en notación decimal y lo convierte a binario.

   - Debe implementarse una función. En caso el número pasado como argumento no cumpla con el requisito imprime un mensaje "Por favor inténtelo nuevamente". En caso no se ingrese un número para la conversión, por defecto se calcula la conversión de 8.
   - Cada dígito de la conversión es almacenado en una lista.
   - La conversión es impresa de forma inversa, esto facilita la solución. Por ejemplo, 6 en binario es 110, pero se almacenaría como [0, 1, 1].
   - **TIP:** usar la conversión por división y *while*.

   Algunos ejemplos de diálogo de este programa serían:

```
Input: conversion()
Output: [0, 0, 0, 1]
```

```
Input: conversion(3)
Output: [1, 1]
```

```
Input: conversion(9)
Output: Por favor inténtelo nuevamente
```

```
Input: conversion(8)
Output: [0, 0, 0, 1]
```

7. (Nivel 3) Implemente una función que reciba un número como parámetro, y haga los cálculos de la siguiente forma:

   - Si el número es múltiplo de 3 entonces la función debe calcular y retornar la suma de todos los números múltiplos de 3 desde el 0 hasta el número pasado como parámetro.
   - Si el número es múltiplo de 4 entonces la función debe calcular y retornar la suma de todos los números múltiplos de 4 desde el 0 hasta el número pasado como parámetro.
   - Si el número es múltiplo de 3 y 4 entonces la función debe retornar la suma de los dos cálculos anteriores
   - Si el número no es múltiplo de 3 ni de 4, o no se le pasa ningun párametro, debe retornar el valor de 0.

   Luego implemente un programa que utilize la función anterior, solicitando al usuario que ingrese un número e imprimiendo el resultado.

   Algunos ejemplos de diálogo de este programa serían:

```
Input: 16
Output: 40
```

```
Input: 9
Output: 18
```

```
Input: 12
Output: 54
```

```
Input: 17
Output: 0
```

8. (Nivel 4) Diseñe e implemente una función que se llame **contar_omitidos** que reciba como parámetro una lista de caracteres ordenados de forma creciente (de 'a' hasta 'z') continuas o separadas por 1 o más caracteres. La función deberá contar cuantas letras han sido omitidas para que la secuencia de letras sea contínua.

   Algunos ejemplos de diálogo de este programa serían:

```
print(contar_omitidos (['a', 'd', 'e', 'i']))
omitidos: 5
```

```
print(contar_omitidos (['a', 'f', 'p']))
omitidos: 13
```

```
print(contar_omitidos (['f', 'p']))
omitidos: 9
```

9. (Nivel 4) Desarrolle un programa que verifique si el número o texto ingresado por el usuario es palíndromo o capicúa.

   Tener en cuenta que se requiere la creación explícita de dos funciones:

   - es_palindromo, el cual recibe como parámetro el número o cadena y retorna verdadero si es palíndromo o falso, en caso contrario.
   - es_capicua, el cual recibe como parámetro el número o cadena y retorna verdadero si es capicúa o falso, en caso contrario.

   Nota: Un *palíndromo* es una palabra que se lee igual hacia delante y hacia atrás y un *capicúa* es un número que se lee igual hacia delante y hacia atrás.

   Considerar los siguientes puntos:

   - El programa lee un número o un texto del usuario.
   - Sí el número es capicúa imprimir:
     - *"número"* es un número capicúa
   - Sí el número no es capicúa imprimir:
     - *"número"* no es un número capicúa

- Sí el texto es palíndromo, imprimir:
  - *"texto"* es un texto palíndromo
- Sí el texto no es palíndromo, imprimir:
  - *"texto"* no es un texto palíndromo

Algunos ejemplos de diálogo de este programa serían:

```
Ingrese un número o una cadena: 23456
23456 no es número capicúa.
```

```
Ingrese un número o una cadena: 123454321
123454321 es un número capicúa.
```

```
Ingrese un número o una cadena: peruano
peruano no es un texto palíndromo.
```

```
Ingrese un número o una cadena: reconocer
reconocer es un texto palíndromo.
```

Listas

1. (Nivel 1) Implemente un algoritmo que permita mostrar los elementos de una lista intercalando uno del inicio con uno del final.

   Algunos ejemplos de diálogo de este programa serían:

```
Lista: [1, 2, 3, 4, 5, 6, 7, 8, 9]
Output: 1 9 2 8 3 4 6 5
```

```
Lista: [2, 2, 2, 3, 3, 3]
Output: 2 3 2 3 2 3
```

2. (Nivel 1) Dada una matriz de valores:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

   Se define la determinante como $D = ad - bc$

   Construya una función en Python para obtener la determinante:

   - Escriba la función como parte de un archivo módulo aparte: ejemplo determinante.py.
   - Escriba un programa main.py en donde se lea los valores *a, b, c, d* y devuelva el resultado de su determinante.

- No olvide invocar con import el módulo de su función.

Algunos ejemplos de ejecución del programa:

```
input:
3
4
5
6
output:
-2
```

```
input:
10
3
7
8
output:
59
```

3. (Nivel 1) Implemente un algoritmo que permita identificar si la suma de los elementos pares de una lista es mayor, menor o igual que la suma de los elementos impares. Si la suma de los pares es mayor, mostrar un mensaje que diga PARES. Si la suma de los elementos impares es mayor, mostrar un mensaje que diga IMPARES. Si son iguales, mostrar un mensaje que diga IGUALES. Asuma que la lista ya se encuentra con elementos.

   Algunos ejemplos de diálogo de este programa serían:

```
lista = [2, 2, 2, 2, 1]
Output: PARES
```

```
lista = [2, 2, 5, 5, 1, 2, 4, 5]
Output: IMPARES
```

```
lista = [3, 2, 3, 2, 2]
Output: IGUALES
```

4. (Nivel 1) Deberás hacer un programa que reciba números hasta que el usuario ingrese el número 0. El programa almacenará esos números en alguna de las siguientes tres listas, y finalmente imprimirá las listas.

   - Los números menores a 100 y pares en una lista.
   - Los números mayores a 100 e impares en otra lista.
   - Los demás números en una tercera lista.

Algunos ejemplos de diálogo de este programa serían:

```
Ingrese un número: 1
Ingrese un número: 2
Ingrese un número: 3
Ingrese un número: 4
Ingrese un número: 5
Ingrese un número: 6
Ingrese un número: 7
Ingrese un número: 8
Ingrese un número: 0

Lista 1: [2, 4, 6, 8]
Lista 2: []
Lista 3: [1, 3, 5, 7]
```

```
Ingrese un número: 142
Ingrese un número: 233
Ingrese un número: 365
Ingrese un número: 4245
Ingrese un número: 5123
Ingrese un número: 63
Ingrese un número: 72
Ingrese un número: 88
Ingrese un número: 34
Ingrese un número: 0

Lista 1: [72, 88, 34]
Lista 2: [233, 365, 4245, 5123]
Lista 3: [63]
```

5. (Nivel 2) Implementar un algoritmo que reciba números enteros hasta que se ingrese el valor cero (0) y los guarde en una lista.

   - Imprima la lista.
   - Crear una lista con todos los números que sean múltiplos de 3.
   - Imprimir el menor número que sea múltiplo de 3.
   - Imprimir la suma de todos los números múltiplos de 3.
   - En caso no existan múltiplos de 3, debe imprimir el primer elemento de la lista original y la suma debe ser 0.

   Algunos ejemplos de diálogo de este programa serían:

```
Ingrese número: 6
Ingrese número: 3456
```

```
Ingrese número: 34
Ingrese número: 0

Lista: [6,3456,34]
Múltiplos de tres: [6,3456]
Menor múltiplo de tres: 6
Suma de múltiplos de 3: 3462
```

```
Ingrese número: 5
Ingrese número: 2
Ingrese número: 4
Ingrese número: 0

Lista: [5,2,4]
Múltiplos de tres: []
Menor múltiplo de tres: 5
Suma de múltiplos de 3: 0
```

6. (Nivel 2) Implementar un algoritmo que imprima el elemento que se repite más veces en una lista, de la siguiente forma:

   - El programa debe pedir al usuario que ingrese 10 valores y se deben guardar en una lista; los valores pueden ser de cualquier tipo de dato,
   - Luego se debe contar las veces que aparece cada elemento en la lista, y
   - Se debe imprimir el elemento que se repite más veces.

   Algunos ejemplos de diálogo de este programa serían:

```
Input: hola
Input: 5
Input: mundo
Input: 78
Input: hola
Input: como
Input: 58
Input: estan
Input: 578
Input: hola
Output: hola se repite 3 veces
```

```
Input: maria
Input: jose
Input: arturo
Input: 15
Input: 25
```

```
Input: 15
Input: manuel
Input: raul
Input: 25
Input: 25
Output: 25 se repite 4 veces
```

7. (Nivel 4) El *coeficiente de Jaccard* es usado para medir el factor de similitud entre dos conjuntos en términos de la intersección y la unión de ambos conjuntos. Se le pide diseñar e implementar un algoritmo que reciba como datos de entrada dos listas A y B, y luego proceder a calcular lo siguiente:

   - $|A \cap B|$: la cantidad de elementos coincidentes en ambos conjuntos.
   - $|A \cup B|$: la cantidad de elementos en ambos conjuntos sin repeticiones.
   - Finalmente, obtener el factor de similitud aplicando el *coeficiente de Jaccard*:

   $$|A \cap B|/|A \cup B|$$

   Nota: Asumir que cada conjunto A y B no contienen elementos repetidos.

   Algunos ejemplos de diálogo de este programa serían:

```
A: 6,5,8,9,7,1
B: 5,2,7,6,8,1

Elementos en la intersección:  5
Elementos en la unión:  7
Factor de similitud Jaccard:  0.71
```

```
A: 7,8,2
B: 11,2,1,7,8,4

Elementos en la intersección:  3
Elementos en la union:  6
Factor de similitud Jaccard:  0.5
```

8. (Nivel 4) Implementar un algoritmo que imprima el número que se repite más veces en una lista y sea múltiplo de 5, de la siguiente forma:

   - El programa debe pedir al usuario que ingrese 10 números y se deben guardar en una lista,
   - Luego se debe contar las veces que aparece cada elemento en la lista y que sea múltiplo de 5, y
   - Se debe imprimir el número que se repite más veces y que sea múltiplo de 5.

Algunos ejemplos de diálogo de este programa serían:

```
Input: 25
Input: 5
Input: 35
Input: 75
Input: 15
Input: 25
Input: 35
Input: 35
Input: 75
Input: 15
Output: 35 se repite 3 veces
```

```
Input: 22
Input: 15
Input: 22
Input: 75
Input: 15
Input: 22
Input: 22
Input: 15
Input: 22
Input: 15
Output: 15 se repite 4 veces
```

9. (Nivel 5) Escribir un programa que tenga como entrada dos números enteros positivos **A** y **B** donde (**A** < **B**) y que tenga como salida todos los números primos cuya suma de sus dígitos sea un número par que existe entre **A** y **B**.

   Considerar los siguientes puntos:

   - Usar listas para manejar los números desde **A** y **B**
   - Si el segundo número ingresado es menor que el primer número ingresado entonces el programa imprime el siguiente mensaje *"A tiene que ser menor que B"* y se seguirá pidiendo los números **A** y **B** hasta que se cumpla esta condición.
   - El resultado debe mostrar a los números primos cuya suma de sus dígitos sea un número par horizontalmente espaciados.

   Algunos ejemplos de diálogo de este programa serían:

```
Ingrese el primer número: 1
Ingrese el segundo número: 10

Resultado: 2
```

```
Ingrese el primer número: 20
Ingrese el segundo número: 10
A tiene que ser menor que B
Ingrese el primer número: 10
Ingrese el segundo número: 20

Resultado: 11 13 17 19
```

```
Ingrese el primer número: 50
Ingrese el segundo número: 100

Resultado: 53 59 71 73 79 97
```

```
Ingrese el primer número: 100
Ingrese el segundo número: 105

Resultado: 101 103
```

Strings (iteraciones y operadores)

1. (Nivel 1) Implemente un algoritmo que permita generar un string a partir de otro (no solo mostrarlo en la consola) de acuerdo a las siguientes reglas:

   - Las mayúsculas se deben convertir en minúsculas.
   - Las minúsculas se deben convertir en mayúsculas.
   - Los números y espacios se deben mantener.
   - Cualquier otro carácter debe ser reemplazado por @.

   Algunos ejemplos de diálogo de este programa serían:

```
Input: Esta vez si sacare 20
Output: eSTA VEZ SI SACARE 20
```

```
Input: UTEC, mi universidad!
Output:utec@ MI UNIVERSIDAD@
```

2. (Nivel 1) Diseñar e implementar un algoritmo que reciba como dato de entrada un texto y proceda a convertir la primera letra de cada palabra a mayúscula y el resto de letras a minúsculas.

   Algunos ejemplos de diálogo de este programa serían:

```
Input: Hola UTECino hoy aprobamos
Output: Hola Utecino Hoy Aprobamos
```

```
Input: ARRIBA PERU
Output: Arriba Peru
```

3. (Nivel 1) Implemente un algoritmo que permita generar un string a partir de otro intercalando una palabra en mayúsculas y otra en mínusculas. Cualquier otro carácter se debe mantener. Asuma que no se combinarán letras y números en una misma palabra. Asimismo, asuma que solo habrá un espacio entre palabra y palabra.

   Algunos ejemplos de diálogo de este programa serían:

```
Input: Hola Juan como estas
Output: HOLA juan COMO estas
```

```
Input: tengo que aprobar el curso sacare 20
Output: TENGO que APROBAR el CURSO sacare 20
```

4. (Nivel 1) Implemente un algoritmo que permita generar un string a partir de otro intercalando una palabra en mayúsculas y otra en mínúsculas. Adicionalmente reemplace la ocurrencia de las vocales por asteriscos.

   Algunos ejemplos de diálogo de este programa serían:

```
Input: Hola Juan como estas
Output: H*L* j**n C*M* *st*s
```

```
Input: Peru pais de todas las sangres
Output: P*R* p**s D* t*d*s L*S s*ngr*s
```

5. (Nivel 2) Crear un programa que deberá leer una frase y luego generará e imprimirá tres cadenas de caracteres con las siguientes características:

   - La primera cadena tendrá todas las palabras de la frase que empiecen con vocal.
   - La segunda cadena tendrá todas las palabras de la frase que empiecen con consonante.
   - La tercera cadena tendrá todas las palabras de la frase, pero al revés.

   Algunos ejemplos de diálogo de este programa serían:

```
Ingrese la frase: La programación es divertida y estimulante

Cadena 1: es estimulante
Cadena 2: La programación divertida y
Cadena 3: etnalumitse y aditrevid se nóicamargorp aL
```

6. (Nivel 3) Implementar un algoritmo que reciba una frase y cuente cuántas letras minúsculas, mayúsculas, números y caracteres especiales (todos los demás caracteres, ejemplo: ,@, , etc.) tiene. Debe:

   - Imprimir la frase.
   - Imprimir la cuenta de letras, números y caracteres especiales.
   - Imprimir la suma de números.
   - Crear e imprimir un string solo con los números.

   Algunos ejemplos de diálogo de este programa serían:

```
Ingrese frase: "Hola␣me␣gust@␣programar␣en␣ICC␣1.07"

Frase: "Hola␣me␣gust@␣programar␣en␣ICC␣1.07"
Mayúsculas: 4
Minúsculas: 20
Caracteres especiales: 8
Suma números: 8
String números: 107
```

```
Ingrese frase: "Me␣sacare␣20␣o␣19␣en␣esta␣pr@ctica"

Frase: "Me␣sacare␣20␣o␣19␣en␣esta␣pr@ctica"
Mayúsculas: 1
Minúsculas: 21
Caracteres especiales: 8
Suma números: 12
String números: 2019
```

7. (Nivel 3) Escribir un programa que reciba como input un número de tamaño mínimo 20 dígitos y que tenga como salida lo siguiente:

   - 1.- La suma del primer y último dígito
   - 2.- Escribir todos los dígitos separados por “,”
   - 3.- Pintar solo los dígitos pares separados por “*”
   - 4.- Imprimir la mitad de los números separados por “-” y la otra mitad separados por “+”

   Algunos ejemplos de diálogo de este programa serían:

```
Ingrese un número de 20 dígitos: 652314578965412365214
1.- Suma del primer y último dígito es: 10
2.- Escribir todos los dígitos separados por ',':
   6,5,2,3,1,4,5,7,8,9,6,5,4,1,2,3,6,5,2,1,4
```

```
3.- Pintar solo los dígitos pares separados por '*':
   6*2*4*8*6*4*2*6*2*4
4.- Imprimir la mitad de los números separados por '-' y la
   otra mitad separados por '+': 6-5-2-3-1-4-5-7-8-9-6
   5+4+1+2+3+6+5+2+1+4
```

```
Ingrese un número de 20 dígitos: 00000000001111111111
1.- Suma del primer y último dígito es: 1
2.- Escribir todos los dígitos separados por ',':
   0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1
3.- Pintar solo los dígitos paress separados por '*':
   0*0*0*0*0*0*0*0*0*0
4.- Imprimir la mitad de los números separados por '-' y la
   otra mitad separados por '+': 0-0-0-0-0-0-0-0-0-0-1
   1+1+1+1+1+1+1+1+1+1
```

```
Ingrese un numero de 20 dígitos:
   98563214754852145632145896547 8
1.- Suma del primer y último digito es: 17
2.- Escribir todos los dígitos separados por ',':
   9,8,5,6,3,2,1,4,7,5,4,8,5,2,1,4,5,6,3,2,1,4,5,8,9,6,5,
   4,7,8
3.- Pintar solo los dígitos pares separados por '*':
   8*6*2*4*4*8*2*4*6*2*4*8*6*4*8
4.- Imprimir la mitad de los números separados por '-' y la
   otra mitad separados por '+':
   9-8-5-6-3-2-1-4-7-5-4-8-5-2-1-4
   5+6+3+2+1+4+5+8+9+6+5+4+7+8
```

8. (Nivel 5) Implemente un algoritmo para formar una nueva cadena con las siguiente reglas:

   - Solicite al usuario que ingrese 2 cadenas, las cadenas pueden contener letras, espacios o números,
   - Para cada cadena ingresada, reemplaze los números por el caracter numeral # ,
   - Convierta las dos cadenas ingresadas a minúsculas,
   - Cree una nueva cadena solo con los caracteres en común de las cadenas ingresadas, sin que se repitan los caracteres en la nueva cadena.

   Algunos ejemplos de diálogo de este programa serían:

```
Cadena 1: hola mundo 123
Cadena 2: como estan 456
Output: oa mn#
```

```
Cadena 1: mi edad es 34
Cadena 2: mi altura es 180
Output: mi eas#
```

```
Cadena 1: 876 Av Franco
Cadena 2: 345 Jr Franco
Output: # afrnco
```

9. (Nivel 5) Implemente un algoritmo para formar una nueva cadena con las siguientes reglas:

   - Solicite al usuario que ingrese 2 cadenas; las cadenas pueden contener letras, espacios o números.
   - Para cada cadena ingresada, reemplaze las vocales con números de la siguiente forma:

     a=1, e=2, i=3, o=4, u=5,
   - Convierta las dos cadenas ingresadas a minúsculas,
   - Cree una nueva cadena solo con los caracteres en común de las cadenas ingresadas, sin que se repitan los caracteres en la nueva cadena.

   Algunos ejemplos de diálogo de este programa serían:

```
Cadena 1: hola mundo 123
Cadena 2: como estan 456
Output: 41 m5n2
```

```
Cadena 1: mi edad es 34
Cadena 2: mi altura es 180
Output: m3 21s
```

```
Cadena 1: Salida
Cadena 2: Entrada
Output: 1d
```

Matrices

1. (Nivel 1) Dada la siguiente matriz de países y medallas:

País	Oro	Plata	Bronce
Estados Unidos	400	350	290
Rusia	390	320	280
Inglaterra	260	330	270
China	230	300	260
Alemania	220	310	270

Implemente un algoritmo que realice lo siguiente:

- Inicialize la matriz de datos en el programa principal,
- Calcule e imprima el total de cada medalla.
- Calcule e imprima la medalla que ha sido repartida más veces.

La salida de este programa sería:

```
Oro:  1500
Plata:  1610
Bronce:  1370
La medalla más repartida fue: Plata
```

2. (Nivel 1) Implemente un algoritmo que permita hallar el menor elemento de una matriz de números. Considere que la matriz puede ser de cualquier tamaño.

   Algunos ejemplos de diálogo de este programa serían:

```
matriz = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
Output: 1
```

```
matriz = [[2, 4], [3, 0]]
Output: 0
```

3. (Nivel 1) Dado un sistema de ecuaciones lineales, genere una matriz con los coeficientes y calcule la diferencia entre los coeficientes de la tercera y la primera ecuación.

$$\begin{cases} -3x \quad +5z = 4 & (a) \\ -2x - y + 5z = 7 & (b) \\ \ \ 2x + y \quad\ \ = 10 & (c) \end{cases}$$

   Algunos ejemplos de diálogo de este programa serían:

```
Matriz: [[-3, 0, 5], [-2, -1, 5], [2, 1, 0]]
Output: [5, 1, -5]
```

4. (Nivel 1) El programa recibirá 50 números, los cuales serán guardados en dos matrices de 5 x 5 (para ello deberá utilizar listas de dos dimensiones). Luego, se deberá crear una tercera matriz con la suma de las dos matrices. Finalmente, el programa creará una matriz con la resta de las dos matrices.

   Algunos ejemplos de diálogo de salida de este programa serían:

```
Salidas:
Matriz Suma:
[[12, 43, 32, 243, 234],
 [98, 54, 13414, 224, 476],
```

```
 [133, 65, 46, 74, 869],
 [4678, 33, 565, 12, 767],
 [657, 346, 78, 212, 909]]

Matriz Resta:
[[0, 13, -2, -177, 26],
 [24, 36, 346, 122, 46],
 [67, 33, 14, -20, 501],
 [656, 23, 45, 0, 55],
 [87, 42, 22, 100, 199]]
```

5. (Nivel 1) Implemente un algoritmo que permita hallar el promedio de los elementos de una matriz de números. Considere que la matriz puede ser de cualquier tamaño

   Algunos ejemplos de diálogo de este programa serían:

```
matriz = [[1, 1, 1], [2, 2, 2], [3 3 3]]
Output: 2
```

```
matriz = [[1, 1], [2, 2]]
Output: 1.5
```

6. (Nivel 1) Escribir un programa que pida del usuario el número de filas **M**, y el número de columnas **N** y que, por medio de una función llamada *"generar_matriz"*, se imprima la matriz donde todos los elementos de la matriz tengan el valor por defecto establecido en la función.

   Considerar los siguientes puntos:

   - La función *"generar_matriz"* recibe tres parámetros:
     - *filas*, como Número de Filas (Parámetro Obligatorio)
     - *columnas*, como Número de Columnas (Parámetro Obligatorio)
     - *elemento*, con el valor default igual al string *"U"* (Parámetro NO Obligatorio)
   - La función *"generar_matriz"* tiene que ser llamada con los parámetros obligatorios.

   Algunos ejemplos de diálogo de este programa serían:

```
Ingrese el número de Filas: 5
Ingrese el número de Columnas: 6

La matriz con 5 filas y 6 columnas es la siguiente:
U U U U U U
U U U U U U
U U U U U U
U U U U U U
U U U U U U
```

```
Ingrese el número de Filas: 2
Ingrese el número de Columnas: 3

La matriz con 2 filas y 3 columnas es la siguiente:
U U U
U U U
```

```
Ingrese el número de Filas: 6
Ingrese el número de Columnas: 1

La matriz con 6 filas y 1 columnas es la siguiente:
U
U
U
U
U
U
```

```
Ingrese el número de Filas: 1
Ingrese el número de Columnas: 6

La matriz con 1 filas y 6 columnas es la siguiente:
U U U U U U
```

7. (Nivel 2) Dada una tabla estadística en donde se ha registrado la incidencia de n enfermedades en m ciudades, cada celda (i, j) de la tabla corresponde a la cantidad de personas que han presentado la enfermedad i en la ciudad j. Se le pide diseñar dos funciones que hagan lo siguiente:

   1. Una función calcular la enfermedad con la mayor incidencia en todas las ciudades.
   2. Una función calcular la ciudad con la mayor incidencia en todas las enfermedades.

   Inicialize una matriz de datos en el programa principal y luego llame a dichas funciones.

   Ejemplo: Si define la siguiente matriz:

	Gripe	Influenza	Cólera
Ciudad 1	200	460	340
Ciudad 2	620	180	200
Ciudad 3	530	350	250

   Se obtiene:

```
La enfermedad con la mayor incidencia: Gripe
La ciudad con la mayor incidencia: Ciudad 3
```

8. (Nivel 2) Implemente un algoritmo que cree una matriz con las siguientes características:

   - Recibe el número de filas y columnas.
   - Si la matriz es cuadrada, todos los valores de la matriz serían '@'.
   - Si la matriz no es cuadrada, los valores son enteros aleatorios entre 1 y 100. Calcula e imprime la multiplicación de todos los valores.
   - En ambos casos se imprime la matriz.

   Algunos ejemplos de diálogo de este programa serían:

```
Ingrese número de filas: 2
Ingrese número de columnas: 2

La matriz es: [['@', '@'], ['@', '@']]
```

```
Ingrese número de filas: 2
Ingrese número de columnas: 3

La matriz es: [[5, 99, 1], [2, 1, 12]]
La multiplicación de elementos es: 11880
```

9. (Nivel 4) Dada la siguiente matriz, de letras y números:

P	O	P	B
E	4	E	2
R	R	T	3
I	5	F	L
C	L	5	G
A	J	3	0

   Implemente un algoritmo que realice lo siguiente:

   - Inicialize la matriz de datos en el programa principal,
   - Solicite al usuario que ingrese un número de fila y el programa debe concatenar e imprimir los caracteres de la fila ingresada
   - Solicite al usuario que ingrese un carácter (letra o número), y el programa debe imprimir la posición o posiciones (fila, columna) de ese carácter

   Algunos ejemplos de diálogo de este programa serían:

```
Ingrese fila: 2
Fila 2: RRT3
Ingrese caracter: L
Output: (3, 3) (4, 1)
```