UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Introducción a Ciencia de la Computación

Ejercicios para Práctica Calificada 3

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

Diccionarios

1. (Nivel 1) Haciendo uso de un diccionario, implemente un algoritmo que permita determinar cuántas veces se repite cada carácter de un string. Considere que su algoritmo debe considerar cualquier carácter (letras, números, símbolos, etc.) excepto espacios en blanco.

   Algunos ejemplos de diálogo de este programa serían:

```
Input: aaabbaa
Output: a: 5, b: 2
```

```
Input: hola como estas
Output: h: 1, o: 3, l: 1, a: 2, c: 1, m: 1, e:1, s: 2
```

2. (Nivel 2) Implemente un algoritmo que solicite al usuario ingresar las notas de las prácticas calificadas. Se termina de ingresar con un número negativo y debe crear e imprimir un diccionario con los siguientes cálculos:

   - Cantidad de notas,
   - Cantidad de notas mayores e iguales a 11.
   - Cantidad de notas menores a 11
   - Promedio de las notas,

   Algunos ejemplos de diálogo de este programa serían:

```
Nota 1: 12
Nota 2: 10
Nota 3: 6
Nota 4: 15
Nota 5: 11
Nota 6: -1

Cantidad Notas: 5
Mayores a 11: 3
```

```
Menores a 11: 2
Promedio: 10.8
```

3. (Nivel 2) Diseñe e implemente un algoritmo que reciba como dato de entrada un texto, y proceda a extraer todas las palabras del texto asociando la cantidad de veces que se repite dicha palabra en el texto. Use un **diccionario** para guardar cada palabra diferente y las veces que se repite.

   Algunos ejemplos de diálogo de este programa serían:

```
Input: Hola Como Estas hola como estas HOLA
Output: {'hola': 3, 'como': 2, 'estas': 2}
```

```
Input: Cuando cuentes cuentos cuenta cuantos cuentos cuentas
    porque si cuentas cuantos cuentos cuentas sabras cuantos
    cuentos sabes contar
Output: {'cuando': 1, 'cuentes': 1, 'cuentos': 4, 'cuenta':
   1, 'cuantos': 3, 'cuentas': 3, 'porque': 1, 'si': 1, '
   sabras': 1, 'sabes': 1, 'contar': 1}
```

4. (Nivel 2) Crea un programa que permita recibir claves y valores para dos diccionarios, hasta que el usuario ponga como clave la palabra "FIN". Uno de los diccionarios guardará todas las claves y valores cuya clave empiece con vocal. Y en el otro diccionario se guardarán las claves y valores cuya clave empiece con consonante. Finalmente le preguntarás al usuario qué diccionarios desea ver, el usuario escogerá y le mostrarás los valores del diccionario escogido.

   Un ejemplo de este programa sería:

```
clave: mama
valor: virginia
clave: autismo
valor: gerundio
clave: papa
valor: Pepe
clave: olor
valor: putrefacto
clave: miriadas
valor: cadenas
clave: aleluya
valor: amen
clave: FIN
Ingrese V o C (vocal o consonante): V
gerundio
putrefacto
amen
```

5. (Nivel 2) Implemente un algoritmo que cuente vocales y consonantes de la siguiente forma:

   - Solicite al usuario que ingrese una frase.
   - Cuente la cantidad de vocales y de consonantes que hay en la frase.
   - Guarde en un diccionario la cantidad de vocales y de consonantes.
   - Imprima el diccionario

   Algunos ejemplos de diálogo de este programa serían:

```
Input: me gusta programar
Output: {"vocales": 6, "consonantes": 10}
```

```
Input: computacion
Output: {"vocales": 5, "consonantes": 6}
```

6. (Nivel 3) Implementar un algoritmo que permita procesar la tabla del número de días feriados por país.

Colombia	20
Venezuela	14
Ecuador	13
Perú	13
Chile	20
Brasil	14
Argentina	16
Paraguay	12
Uruguay	16
Bolivia	11

   1. Crear un diccionario con la información de la tabla. Los siguientes items deben procesar el diccionario para obtener los resultados.
   2. El usuario ingresa un nombre de país y el programa imprime la cantidad de días feriados si este se encuentra en el diccionario. Caso contrario imprime el mensaje "El nombre del país no se encuentra en nuestros datos".
   3. El usuario ingresa un número de días `N` y el programa imprime todos los países que tienen más de `N` días feriados.
   4. Crear un nuevo diccionario con los países que tienen más de `N` días feriados. Usando el nuevo diccionario calcular e imprimir la suma total de feriados.

   Algunos ejemplos de diálogo de este programa serían:

```
Input:
Por favor ingrese un nombre de país: Perú
Por favor ingrese un numero de días: 19
Output:
Los feriados de Perú son 13
Los países que tienen más de 19 feriados son: Colombia Chile
El total de feriados es: 40
```

```
Input:
Por favor ingrese un nombre de país: perues
Por favor ingrese un numero de días: 100
Output:
El nombre del país no se encuentra en nuestros datos
Los paises que tienen más de 100 feriados son:
El total de feriados es: 0
```

7. (Nivel 4) Dado una cadena de texto llamada 'planetas' que contiene los planetas del sistema solar que incluye la estrella Sol separados por el texto '@planeta$' de la siguiente forma:

   planetas = "Sol@planeta$Mercurio@planeta$Venus@planeta$Tierra@planeta$Marte@planeta

   $Jupyter@planeta$Saturno@planeta$Urano@planeta$Neptuno@planeta$Pluton"

   Se pide resolver lo siguiente:

   - Crear un diccionario con llave entera desde 1 hasta 10 en forma correlativa y valor string que representa el planeta a partir de la cadena 'planetas'
   - Se pide crear una función que reciba como parámetro el índice y retorne el valor. El índice es leído por teclado.
   - Se pide crear una función que reciba como parámetro el valor y retorne el índice. El valor es leído por teclado.

   Nota:

   - Validar que la llave y el valor ingresado sea del tipo de dato correspondiente y en el caso de la llave que este en el rango desde 1 hasta 10.

   Algunos ejemplos de diálogo de este programa serían:

```
Ingrese una llave desde 1 hasta 10: 5
El valor con llave 5 es Marte
Ingrese el nombre de un planeta: tierra
El valor con llave tierra es 4
```

```
Ingrese una llave desde 1 hasta 10: 11
Ingrese una llave desde 1 hasta 10: -1
Ingrese una llave desde 1 hasta 10: 10
El valor con llave 10 es Pluton
Ingrese el nombre de un planeta: 123
Ingrese el nombre de un planeta: planeta
Ingrese el nombre de un planeta: sol
El valor con llave sol es 1
```

```
Ingrese una llave desde 1 hasta 10: 4
El valor con llave 4 es Tierra
Ingrese el nombre de un planeta: sol
El valor con llave sol es 1
```

Recursividad

1. (Nivel 1) Desarrolle un algoritmo que implemente una función **recursiva** que permita contar cuántos dígitos tiene un número.

   Algunos ejemplos de diálogo de este programa serían:

   ```
   Input: 123456
   Output: 6
   ```

   ```
   Input: 111
   Output: 3
   ```

2. (Nivel 1) Elabore un programa recursivo para calcular el número armónico, este algoritmo se define por:

   $$H_n = 1 + \frac{1}{2} + \frac{1}{3} + ..\frac{1}{N}$$

   Esta función recibirá como parámetros los siguientes valores:

   - numero (N): Es el número de la secuencia.

   Tome en consideración la condición de salida y la llamada recursiva.

   Algunos ejemplos de diálogo de este programa serían:

   ```
   Input: 2
   Output: 1.5
   ```

   ```
   Input: 10
   Output: 2.929
   ```

3. (Nivel 2) Diseñe e implemente una **función recursiva** para hallar el mayor elemento de una lista de números. ¿Cuál sería el caso base? ¿Cuál sería el caso recursivo?

   Algunos ejemplos de diálogo de este programa serían:

```
Input: 5, 6, 2, 3, 8, 1, 5
Output: 8
```

```
Input: 1, 2, 3, 4, 5, 6
Output: 6
```

4. (Nivel 2) Realiza un programa que reciba dos números enteros y realice la división de ambos, pero de forma recursiva.

   - En caso el segundo número sea 0, deberá mostrar un mensaje explicando que no se puede dividir entre cero.
   - El programa no permitirá números mayores a 10500.

   Algunos ejemplos de diálogo de este programa serían:

```
Input:
50
5
Output:
10
```

```
Input:
120
30
Output:
4
```

5. (Nivel 2) Generar una lista de números enteros usando comprensión y que se encuentre el menor y el mayor elemento ingresado a la lista utilizando recursividad.

   Tener en cuenta lo siguiente:

   - Debe crear la función recursiva mayor.
   - Debe crear la función recursiva menor.
   - Pedir al usuario el número de elementos a ingresar.
   - Pedir al usuario que ingrese cada elemento correlativamente.

   Algunos ejemplos de diálogo de este programa serían:

```
Ingrese el número de elementos de la lista: 5
Ingrese elemento #1: 10
Ingrese elemento #2: 11
```

```
Ingrese elemento #3: 78
Ingrese elemento #4: 16
Ingrese elemento #5: 96
El menor elemento de la lista es 10
El mayor elemento de la lista es 96
```

```
Ingrese el número de elementos de la lista: 3
Ingrese elemento #1: 11523
Ingrese elemento #2: 10
Ingrese elemento #3: 14584
El menor elemento de la lista es 10
El mayor elemento de la lista es 14584
```

```
Ingrese el número de elementos de la lista: 10
Ingrese elemento #1: 1
Ingrese elemento #2: 9
Ingrese elemento #3: 8
Ingrese elemento #4: 7
Ingrese elemento #5: 5
Ingrese elemento #6: 6
Ingrese elemento #7: 15
Ingrese elemento #8: 4
Ingrese elemento #9: 12
Ingrese elemento #10: 14
El menor elemento de la lista es 1
El mayor elemento de la lista es 15
```

6. (Nivel 2) Desarrolle un algoritmo que implemente una función **recursiva** que permita contar cuántos dígitos pares hay en un número entero. Considere que el cero es par.

   Algunos ejemplos de diálogo de este programa serían:

```
Input: 123456
Output: 3
```

```
Input: 111
Output: 0
```

```
Input: 2220
Output: 4
```

7. (Nivel 2) Implemente un algoritmo, usando una función recursiva, que resuelva la siguiente sumatoria:

$$H(n,d) = 1/d + 2/d + 3/d + ... + N/d$$

   - El programa debe solicitar al usuario que ingrese un numero n, y un numero d,

- Luego debe calcular el valor de $H(n, d)$ usando una función recursiva,
- Debe imprimir el resultado de $H(n, d)$

Algunos ejemplos de diálogo de este programa serían:

```
Input n: 3
Input d: 2
Output: 3
```

```
Input n: 5
Input d: 5
Output: 3
```

8. (Nivel 3) Implemente un algoritmo, usando una función recursiva, que resuelva la siguiente sumatoria:

$$K(n, p) = p + 2 * p + 3 * p + 4 * p + ... + n * p$$

- El programa debe solicitar al usuario que ingrese un número n, y un número d,
- Luego debe calcular el valor de $H(n, p)$ usando una función recursiva,
- Debe imprimir el resultado de $H(n, p)$

Algunos ejemplos de diálogo de este programa serían:

```
Input n: 5
Input p: 2
Output: 30
```

```
Input n: 6
Input p: 3
Output: 63
```

9. (Nivel 5) Implementar un algoritmo que permita calcular el MCD (Máximo Común Divisor) de dos números.

- Implementar una función recursiva que calcule el MCD.
- El usuario ingresa dos números enteros por teclado.
- Imprimir el resultado del MCD.

Algunos ejemplos de diálogo de este programa serían:

```
Input:
Por favor ingrese número A: 12
Por favor ingrese número B: 3
Output:
El máximo común divisor es: 3
```

```
Input:
Por favor ingrese número A: 169
Por favor ingrese número B: 256
Output:
El máximo común divisor es: 1
```

Matriz 2 dimensiones

1. (Nivel 1) Dado un sistema de ecuaciones lineales. Genere una matriz con los coeficientes y calcule la suma entre los coeficientes de la tercera ecuación y la primera. Considere que el usuario ingresa los valores de los coeficientes.

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

   Algunos ejemplos de diálogo de este programa serían:

```
Input:
Ingrese fila1 columa1: -3
Ingrese fila1 columa2: 0
...

Output: [-1, 1, 5]
```

2. (Nivel 3) Se dice que una matriz es izquierda si la suma de los elementos que se encuentran en las columnas pertenecientes a la mitad izquierda de la matriz es mayor a la mitad derecha, y derecha si es al revés. En caso sean iguales, se dice que es una matriz neutra. Considere que si el número de filas es impar, la columna del medio deberá contar para la parte izquierda. Desarrolle un algoritmo que permita determinar si una matriz es izquierda, derecha o neutra.

   Algunos ejemplos de diálogo de este programa serían:

```
Input: [[0, 1], [0, 1]]
Output: Derecha
```

```
Input: [[2, 1, 1], [2, 1, 1]
Output: Izquierda
```

```
Input: [[1, 1], [1, 1]]
Output: Neutra
```

3. (Nivel 3) Desarrolle un programa que generará 5 matrices de 4 filas y 4 columnas con números aleatorios entre 1 y 100. Luego el programa escogerá aquellas dos matrices cuya suma de números sea mayor e imprimirá la matriz, también de 4 x 4, con la suma de ambas.

   Algunos ejemplos de diálogo de este programa serían:

```
La matriz cuadrada generada es:
[[132, 172, 157, 141],
 [156, 151, 183, 138],
 [116, 167, 110, 127],
 [96, 119, 162, 119]]
```

4. (Nivel 3) Desarrollar un programa que genere una matriz cuadrada de lado igual a N (ingresado por el usuario), donde la diagonal principal sean solo 1's y el resto de la matriz, números aleatorios entre 1 y 10. Además, se pide crear una función que reciba como parámetro la matriz y retorne su transpuesta.

   Transpuesta: La transpuesta de una matriz es una operación que cambia filas por columnas:

   Ejemplo

   Dado la matriz:

$$\begin{aligned} A \;=\; & [[2,4], \\ & [6,8], \\ & [10,12]] \end{aligned}$$

   La transpuesta de la Matriz A es:

$$\begin{aligned} A\_transpuesta \;=\; & [[2,6,10], \\ & [4,8,12]] \end{aligned}$$

   La fila de la matriz A ahora es la columna de la matriz A_transpuesta

   Algunos ejemplos de diálogo de este programa serían:

```
Ingrese el lado de la matriz cuadrada: 5
La matriz cuadrada generada es:
[[1, 2, 5, 1, 4],
 [4, 1, 3, 8, 2],
 [6, 7, 1, 7, 6],
 [6, 9, 6, 1, 1],
 [10, 5, 1, 10, 1]]

La transpuesta de la matriz es:
[[1, 4, 6, 6, 10],
 [2, 1, 7, 9, 5],
```

```
 [5, 3, 1, 6, 1],
 [1, 8, 7, 1, 10],
 [4, 2, 6, 1, 1]]
```

```
Ingrese el lado de la matriz cuadrada: 6
La matriz cuadrada generada es:
[[1, 2, 1, 6, 9, 7],
 [6, 1, 5, 9, 3, 4],
 [7, 6, 1, 8, 7, 8],
 [3, 6, 8, 1, 2, 5],
 [5, 4, 8, 9, 1, 2],
 [7, 10, 10, 9, 7, 1]]

La transpuesta de la matriz es:
[[1, 6, 7, 3, 5, 7],
 [2, 1, 6, 6, 4, 10],
 [1, 5, 1, 8, 8, 10],
 [6, 9, 8, 1, 9, 9],
 [9, 3, 7, 2, 1, 7],
 [7, 4, 8, 5, 2, 1]]
```

```
Ingrese el lado de la matriz cuadrada: 7
La matriz cuadrada generada es:
[[1, 2, 1, 3, 10, 6, 5],
 [10, 1, 8, 5, 8, 5, 1],
 [3, 7, 1, 7, 10, 6, 10],
 [6, 10, 6, 1, 7, 2, 3],
 [9, 2, 2, 2, 1, 5, 10],
 [8, 9, 7, 10, 2, 1, 6],
 [7, 10, 1, 6, 10, 7, 1]]

La transpuesta de la matriz es:
[[1, 10, 3, 6, 9, 8, 7],
 [2, 1, 7, 10, 2, 9, 10],
 [1, 8, 1, 6, 2, 7, 1],
 [3, 5, 7, 1, 2, 10, 6],
 [10, 8, 10, 7, 1, 2, 10],
 [6, 5, 6, 2, 5, 1, 7],
 [5, 1, 10, 3, 10, 6, 1]]
```

5. (Nivel 3) Se dice que una matriz es superior si la suma de los elementos que se encuentran en las filas pertenecientes a la mitad superior de la matriz es mayor a la mitad inferior e

inferior si es al revés. En caso sean iguales, se dice que es una matriz neutra. Considere que si el número de filas es impar, la fila del medio deberá contar para la parte superior. Desarrolle un algoritmo que permita determinar si una matriz es superior, inferior o neutra.

Algunos ejemplos de diálogo de este programa serían:

```
Input: [[1, 1, 1], [1, 1, 1], [1, 1, 1]]
Output: Superior
```

```
Input: [[1, 2, 3], [1, 2, 3]]
Output: Neutra
```

```
Input: [[2, 2], [1, 1], [8, 8]]
Output: Inferior
```

6. (Nivel 4) Dada la siguiente matriz de números:

4	5	6	3
1	4	6	7
8	2	3	6
3	5	4	9
3	4	5	3

Implemente un algoritmo que realice lo siguiente:

- Escribir la matriz en el programa principal
- Solicite al usuario un número de columna, y el programa debe imprimir el promedio de los números de esa columna
- Solicite al usuario un número de fila y columna, y el programa debe:

  Imprimir el producto de todos los números que están alrededor de esa posición de fila y columna.

  Imprimir el mayor número que está alrededor de esa posición de fila y columna

Recordar que la fila y columna comienzan desde cero.

Algunos ejemplos de diálogo de este programa serían:

```
Ingrese columna: 3
El promedio de la columna 3 es: 5.6
Ingrese fila: 4
Ingrese columna: 1
El producto alrededor de (4, 1) es: 900
El mayor numero alrededor de (4, 1) es: 5
```

7. (Nivel 4) Diseñe e implemente un función que reciba como parámetro una matriz de datos y proceda a realizar la rotación de 90 grados en sentido horario y el resultado retornarlo en una matriz de salida.

   Algunos ejemplos de entrada y salida de esta función serían:

```
Input:
5 4 2
1 2 3
0 6 7

Output:
0 1 5
6 2 4
7 3 2
```

```
Input:
1 1 1 1 1 1
1 1 1 1 1 1
0 0 1 1 0 0
0 0 1 1 0 0
0 0 1 1 0 0

Output:
0 0 0 1 1
0 0 0 1 1
1 1 1 1 1
1 1 1 1 1
0 0 0 1 1
0 0 0 1 1
```

8. (Nivel 4) Implementar un algoritmo que permita realizar la siguiente operación con matrices. **TIP:** Puede usar loops anidados o *comprehension*.

   1. El usuario ingresa un número `N`, correspondiente al número de filas y columnas (matrices cuadradas).
   2. Implementar una función que reciba como argumento `N` y cree una matriz cuadrada con valores enteros aleatorios entre 1 y 10. Crear dos matrices A y B usando esta función.
   3. Implementar una función que reciba como argumentos dos matrices y realice la suma de estas. Calcular la suma de las matrices A y B y guardar el resultado en una matriz C.
   4. Implementar una función que reciba como argumento una matriz y la imprima (ver formato en ejemplo). Imprimir las matrices A, B y C usando esta función.

   Algunos ejemplos de diálogo de este programa serían:

```
Por favor ingrese N: 2

A=
1 3
2 8
B=
4 1
7 2
C=
5 4
9 10
```

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

4	5	6	3
8	2	3	6
3	5	4	9
1	4	6	7
3	4	5	3
2	3	3	4

Implemente un algoritmo que realice lo siguiente:

- Escribir la matriz en el programa principal
- Solicite al usuario un número de columna, y el programa debe imprimir la suma de todos los números de esa columna
- Solicite al usuario un número de fila y columna, y el programa debe:

  Calcular e imprimir la suma de todos los números que están alrededor de esa posición de fila y columna.

  Calcula e imprimir el menor número que está alrededor de esa posición de fila y columna

Algunos ejemplos de diálogo de este programa serían:

```
Ingrese columna: 2
La suma de la columna 2 es: 27
Ingrese fila: 2
Ingrese columna: 0
La suma alrededor de (2, 0) es: 20
El menor número alrededor de (2, 0) es: 1
```

Complejidad

1. (Nivel 1) Dado el siguiente algoritmo:

```
n = int(input("dime un numero"))
for x in range(n):
   for y in range(n):
      for z in range(n):
            print(x,y,z)
```

   Indicar la complejidad algorítmica que posee. Luego comentar cuáles son los posibles problemas de un algoritmos con este tipo de complejidad.

2. (Nivel 2) Dado los siguientes enunciados, elegir la alternativa correcta.

   1. Dado el siguiente algoritmo:

```
def algorithm1(data, value):
    for index in range(len(data)):
        if value == data[index]:
            return index
    raise ValueError("Valor no encontrado en la lista.")
```

      Identificar la complejidad algorítmica:

      (a) $\mathcal{O}(1)$
      (b) $\mathcal{O}(n \log(n))$
      (c) $\mathcal{O}(n)$
      (d) $\mathcal{O}(\log(n))$
      (e) $\mathcal{O}(n^2)$

   2. Dado el siguiente algoritmo:

```
def algorithm2(n):
    if n<=1:
        return 1
    return algorithm2(n-1)*n
```

      Identificar la complejidad algorítmica:

      (a) $\mathcal{O}(1)$
      (b) $\mathcal{O}(n \log(n))$
      (c) $\mathcal{O}(n^2)$
      (d) $\mathcal{O}(log(n))$
      (e) $\mathcal{O}(n!)$

   3. Dado el siguiente algoritmo:

```
def algorithm3(data):
    return data[0]
```

Identificar la complejidad algorítmica:

(a) $\mathcal{O}(1)$
(b) $\mathcal{O}(n \log(n))$
(c) $\mathcal{O}(n^3)$
(d) $\mathcal{O}(log(n))$
(e) $\mathcal{O}(n!)$

4. Dado el siguiente algoritmo:

```
def algorithm4(data, value):
    n = len(data)
    left = 0
    right = n - 1
    while left <= right:
        middle = int((left+right)/2)
        if value < data[middle]:
            right = middle - 1
        elif value > data[middle]:
            left = middle + 1
        else:
            return middle
    raise ValueError("Valor␣no␣esta␣en␣la␣lista")
```

Identificar la complejidad algorítmica:

(a) $\mathcal{O}(log(n))$
(b) $\mathcal{O}(n \log(n))$
(c) $\mathcal{O}(1)$
(d) $\mathcal{O}(n^2)$
(e) $\mathcal{O}(n!)$

5. Dado el siguiente algoritmo:

```
def algorithm5(a, b):
    if a > b:
        return a
    else:
        return b
```

Identificar la complejidad algorítmica:

(a) $\mathcal{O}(log(n))$
(b) $\mathcal{O}(n \log(n))$
(c) $\mathcal{O}(n^3)$
(d) $\mathcal{O}(1)$
(e) $\mathcal{O}(2^n)$

3. (Nivel 2) Analizar la complejidad de los siguientes algoritmos.

    1. En menos de 30 palabras indicar que hace el siguiente código. Indicar cuál es su time complexity en función de n (Ejemplo: $O(n)$, $O(n^2)$, $O(n^3)$, $O(log(n))$, etc.)

```
variable=0
for i in range(n):
    variable+=i
```

    2. En menos de 30 palabras indicar que hace el siguiente código. Indicar cuál es su time complexity en función de n (Ejemplo: $O(n)$, $O(n^2)$, $O(n^3)$, $O(log(n))$, etc.)

```
for i in range(n):
    for j in range(n/2):
        variable+=(i*j)
```

    3. Dibujar un plot simple con las funciones de complejidad correspondientes a cada código. Analizando su gráfico, indicar: ¿cuál de ambos códigos ejecutará en menos tiempo?

4. (Nivel 3) Compare la implementación iterativa de una función factorial con la de una implementación recursiva en términos de tiempo y velocidad. Explique cuál de las dos es más eficiente o si son iguales.

5. (Nivel 3) Compare la implementación iterativa de una función sumatoria con la de una implementación recursiva en términos de tiempo y velocidad. Explique cuál de las dos es más eficiente o si son iguales.

6. (Nivel 3) Para cada una de las siguientes funciones, realize lo siguiente:

    - Escriba qué hace la función,
    - Calcule la complejidad algorítmica de la función,
    - Escriba la interpretación de la complejidad algorítmica calculada.

```
def codigo_1( number ):
    a = 0
    for j in range(1, number+1):
        a += a + j

    for k in range(number, 0, -1):
        a -= 1
        a *= 2
    return a
```

```
def codigo_3( number ):
    a = 0;
```

```
    for j in range(1, number+1):
        for k in range(1, number+1):
            a += a + ( k*j )

     return a
```

7. (Nivel 3) Para cada una de las siguientes funciones, realize lo siguiente:

   - Escriba qué hace la función,
   - Calcule la complejidad algorítmica de la función,
   - Escriba la interpretación de la complejidad algorítmica calculada.

```
def codigo_1( numero ):
    a = 0
    for j in range(1, numero+1):
        a += a + j

    for k in range(numero, 0, -1):
        a -= 1
    return a
```

```
def codigo_2( numero ):
    a = 0

    for j in range(1, numero+1):
        for k in range(1, numero+1):
            a += 1 + ( k*j )

     return a
```

8. (Nivel 3) A continuación se dan dos algoritmos que resuelven el mismo problema y reciben como parámetro una matriz y un elemento. Analice la complejidad computacional de ambos algoritmos (en función de la cantidad de iteraciones) y responda lo siguiente:

   - Indique qué hacen exactamente ambos algoritmos.
   - ¿Cuál de los dos algoritmos es más eficiente? ¿Por qué?.

```
######### Algoritmo 1 ################
def f1(matriz, elemento):
  resultado = False
  for i in range(len(matriz)):
    for j in range(len(matriz[0])):
      if elemento == matriz[i][j]:
        resultado = True
```

```
  return resultado

######### Algoritmo 2 ################
def f2(matriz, elemento):
  resultado = False
  i = 0
  while i < len(matriz) and not resultado:
    j = 0
    while j < len(matriz[0]) and not resultado:
      if elemento == matriz[i][j]:
        resultado = True
      j+=1
    i+=1
  return resultado
```

9. (Nivel 3) Para cada una de las siguientes funciones, realize lo siguiente:

   - Identifique la función $\mathcal{O}(n)$ de crecimiento.
   - Indique que función tomará más tiempo conforme crece la cantidad de datos.

```
def funcion_1( number ):
    acumulador = 0
    for j in range(1, number+1):
        acumulador = acumulador + j

    for k in range(number/2, 0, -1):
        acumulador *= 2
    return acumulador
```

```
def funcion_2( number ):
    a = 0;

    for i in range(1, number+1):
        for j in range(1, number+1):
            a +=( i+j )

     return a
```