PROBLEMA 1 (15 puntos)

a) (12 puntos) Indique que se muestra por pantalla al ejecutar el siguiente código:

```
def misterio(a, b):
#3->
p = 0
#4->
while b > 0:
  b = b - 1
  p = p + a
#5->
return p

a = int(input('Ingrese a'))
#1->
b = int(input('Ingrese b'))
#2->
valor_misterioso = misterio(a, b)
#6->
print('El valor misterioso de a y b es', valor_misterioso)
```

Complete los cuadros de la siguiente tabla de ejecución, que contiene los valores que toma cada variable en los instantes dados **ingresando los valores 5 y 4**. Marque con una X si la variable no tiene valor en algún instante.

				misterio		
Valor en	a	b	valor_misterioso	a	b	p
# 1 →	5	x	x	x	x	x
# 2 →	5	4	x	x	x	x
# 3 →	5	4	x	5	4	0
# 4 →	5	4		5	4,3,2,1,0	0,5,10,15,20
# 5 →	5	4	20	5	0	20
# 6 →	x	x	20	x	x	x

b) (3 puntos) Describa **brevemente** el propósito de la función.

NOTA: Explicar paso a paso lo que hace cada instrucción de la función NO dará puntos.

Solución:

a) El programa muestra por pantalla:

1 punto por cada linea

Ingrese a 5
Ingrese b 4
El valor misterioso de a y b es 20

				misterio		
Valor en	a	b	valor_misterioso	a	b	p
# 1 →	5	X	X	X	X	X
# 2 →	5	4	X	X	X	X
# 3 →	X ó 5	X ó 4	X	5	4	X
# 4 →	X ó 5	X ó 4	X	5	4	0
# 5 →	X ó 5	X ó 4	X	5	0	20
# 6 →	5	4	20	X	X	X

1 punto por cada linea correcta para #1, #2, y #4
2 puntos por cada linea correcta para #3, #5, y #6

b) La función calcula la multiplicación entre dos números sumando a, b veces.

PROBLEMA 2 (20 puntos)

Escriba un programa que permita crear una nueva lista de palabras, solicitando al usuario una cantidad de palabras y cada una de las palabras. Luego, limpie la nueva lista, eliminando todas las palabras repetidas de la lista, debe dejar en la lista la primera aparición de esa palabra. Hint: genere una segunda lista con las palabras que no se repitan. Finalmente, muestre el contenido de la nueva lista.

Un ejemplo de ejecución del programa es el siguiente:

ingrese el número de palabras:
4
ingrese la palabra 1
hoy
ingrese la palabra 2
ayer
ingrese la palabra 3
mañana
ingrese la palabra 4
ayer
La lista original es ['hoy', 'ayer', 'mañana', 'ayer']

La lista modificada es ['hoy','ayer','mañana']

Solución:

```
print("ingrese el número de palabras: ")
num = int(input())
# 2 Puntos

lista = []
for i in range (0,num):
    print("ingrese la palabra ",i+1)
    val = input()
    lista.append(val)
# 3 Puntos

print("La lista original es", lista)
# 1 Punto

nuevaLista=[]  #1 Punto
for i in lista:   #1 Punto
   #10 puntos por la lógica
    flag =1
    for j in nuevaLista: #1 Punto
        if i==j:
            flag=0
    if flag==1:
        nuevaLista.append(i)

print("La lista limpia es", nuevaLista)
# 1 Punto
```

PROBLEMA 3 (25 puntos)
Los profesores de cálculo quieren demostrarle a los alumnos de primer año (¡Ustedes!) que una forma de calcular la raíz cúbica de un número es iterar la siguiente fórmula

$$X = (2 * X^3 + num) / (3 * X^2)$$

hasta que el valor absoluto de X_{actual} - $X_{anterior}$ sea menor que un cierto valor dado que llamaremos epsilon.

Por ejemplo, para calcular la raíz cúbica de 8 con un valor de epsilon de 0.001, al algoritmo repite el cálculo de la fórmula de la siguiente forma:

$X = 1.000$	# X inicial
$X = (2 * 1.000^3 + 8) / (3 * 1.000^2) = 3.333$	# 3.333 - 1.000 = 2.333, que es mayor a epsilon
$X = (2 * 3.333^3 + 8) / (3 * 3.333^2) = 2.462$	# 3.333 - 2.462 = 0.871, que es mayor a epsilon
$X = (2 * 2.462^3 + 8) / (3 * 2.462^2) = 2.081$	# 2.462 - 2.081 = 0.381, que es mayor a epsilon
$X = (2 * 2.081^3 + 8) / (3 * 2.081^2) = 2.003$	# 2.081 - 2.003 = 0.078, que es mayor a epsilon
$X = (2 * 2.003^3 + 8) / (3 * 2.003^2) = 2.000$	# 2.003 - 2.000 = 0.003, que es mayor a epsilon
$X = (2 * 2.000^3 + 8) / (3 * 2.000^2) = 2.000$	# 2.000 - 2.000 = 0, que es menor a epsilon. Entonces termino la ejecución.

Los profesores de cálculo no son buenos programadores, así que nos han solicitado a los profesores de programación que lo hagamos… y nosotros se lo pedimos a Ustedes (que ironía, ¿no?).

Se les pide escribir una función en Python llamada **raiz_cubica** que reciba como parámetro un número **num** y que retorna el valor aproximado de la raíz cúbica de este número utilizando la fórmula anterior. Para esta función, considere un valor de **epsilon igual a 0.001** y un **valor inicial de X_{actual} igual a 1** y un **valor inicial de $X_{anterior}$ igual a 0**.

Recuerda que la librería **math** contiene la función **pow(y,z)** que calcula **y elevado z**, y la función **fabs(num)** que retorna el **valor absoluto de num**

Solución:

```
import math    (1 punto)

def raiz_cubica(num):   (1 definición puntos)
    X=1          (4 inicializa valores)
    Xant=0
    while math.fabs((X-Xant)) >= 0.001:  ( Condición del while correcta 8 puntos)
        Xant=X     (modificación del valor anterior 6 puntos)
        X=(2*math.pow(X,3)+num)/(3*math.pow(X,2))  (función recursiva 4 puntos)
    return X       (1 retorna puntos)
```

Se descuenta 0.3 por no colocar math.

OTRA POSIBLE SOLUCIÓN USANDO LISTAS

```
import math

lista = [0, 1]

def raiz_cubica(num):
    X = lista[-1]
    while math.fabs((X-lista[-2])) >= 0.001:
        lista.append((2 * math.pow(X, 3) + num) / (3 * math.pow(X, 2)))
    return lista[-1]
```