Ciclo 1 - Fundamentos de programación

Reto 2

La entidad Avícola Colombia debe administrar las facturas que tiene por cobrar, por venta de sus productos. Estas facturas son almacenadas en una base de datos en forma de diccionario, en donde la clave es igual al *numero de la factura*, y el valor igual al *valor total de la factura*. Se requiere para la gestión de este proceso un programa que le permita a la entidad añadir nuevas facturas, de igual forma deberá poderse abonar a estas facturas un valor parcial o pagarla completamente, en caso de realizarse un abono por el valor total de la factura, esta deberá ser eliminada. Se asume que nunca se va a realizar un abono superior al valor pendiente. Posterior a la operación de abono o adición de factura, se deberá mostrar por pantalla, como mensaje los valores de el cliente, el numero de factura, la cantidad abonada hasta el momento y el valor pendiente: {'cliente': *idCliente*, 'factura': *numFactura*, 'abono': *valorCalculado*, 'valor': *valorCalculado*}, y el estado de la base de datos. Si la factura sobre la que se desea abonar no existe deberá mostrar como mensaje: {'*numFactura*': 'No existe la factura'}

Cada factura tiene asociado un cliente, antes de poder realizar cualquier operación sobre una factura deberá existir el cliente asociado. Para el cliente solo se almacena *idCliente*. Cada cliente inicia con un diccionario vació. Si trata de operar sobre una factura para un cliente que no existe deberá mostrar como mensaje: {'*idCliente*': 'No existe el cliente'}
Si se hace un llamado para la creación de un cliente y se envian datos de una factura no se podrá crear el cliente y se deberá mostrar el mensaje para cliente que no existe. Al crear un cliente se debe mostrar como mensaje: {'*idCliente*': 'Cliente creado'}.

Tabla 1. Entradas

Nombre	Tipo de Dato	Descripción
opcion	int	valor entero entre 0 y 3
idCliente	int	valor entero mayor a 0
numFactura	int	valor entero mayor a 0
valor	float	valor flotante mayor a 0
db	dict	diccionario con el estado actual de la base de datos

Tabla 2. Salida

Tipo de Retorno	Descripción
dict	{'llaveMensaje': 'mensaje'} {} el primer diccionario contiene el mensaje del sistema el segundo diccionario el estado de la base de datos

Se podrá en cualquier momento solicitar el estado de la base de datos, en este caso el mensaje es {'print': 'estado de la base de datos'}

Para todos los casos se debe mostrar el respectivo mensaje y el estado de la base de datos: {'llaveMensaje': 'mensaje'} {}

Los valores que se pueden enviar como opción son:

0 Crear cliente

1 Añadir factura

2 Abono parcial o total

3 Mostrar base de datos

Esqueleto: Úselo como base para el desarrollo de su solución

```
def facturas(opcion: int, idCliente: int = 0, numFactura: int =
    0, valor: int = 0, db:dict={}) -> dict:
    # Su codigo
    pass
```

Nota: En la plataforma debe cargar solo la función, con el mismo nombre dado, recibiendo los mismos parámetros y respetando la estructura de salida.

Caso de prueba 1: (Consulte las salidas en la Tabla 3)

```
# Pruebas
msj, dbFacturas = facturas(0,
    2541)
print(msj, dbFacturas)
msj, dbFacturas = facturas(1,
    2541, 1, 300000, db=
    dbFacturas)
print(msj, dbFacturas)
msj, dbFacturas = facturas(2,
    2541, 1, 25000.25487, db
    =dbFacturas)
print(msj, dbFacturas)
msj, dbFacturas = facturas(1,
    2541, 2, 500000, db=
    dbFacturas)
print(msj, dbFacturas)
msj, dbFacturas = facturas(2,
    1429, 5, 25000.25487, db
    =dbFacturas)
print(msj, dbFacturas)
msj, dbFacturas = facturas(1,
    1429, 1, 700000, db=
    dbFacturas)
print(msj, dbFacturas)
msj, dbFacturas = facturas(2,
    1429, 1, 700000, db=
    dbFacturas)
print(msj, dbFacturas)
msj, dbFacturas = facturas(0,
    1429, 1, 700000, db=
    dbFacturas)
print(msj, dbFacturas)
msj, dbFacturas = facturas(0,
    1429, db=dbFacturas)
print(msj, dbFacturas)
msj, dbFacturas = facturas(1,
    1429, 1, 700000, db=
    dbFacturas)
print(msj, dbFacturas)
msj, dbFacturas = facturas(2,
    2541, 1, 274999.74513,
    db=dbFacturas)
print(msj, dbFacturas)
msj, dbFacturas = facturas(3,
    db=dbFacturas)
print(msj, dbFacturas)
```

Tabla 3. Entradas y salidas de caso de prueba 1

Entradas					Salida
opcion	idCliente	numFactura	valor	db	return
0	2541				{'2541': 'Cliente creado'} {2541: {}}
1	2541	1	300000	{2541: {}}	{'cliente': 2541, 'factura': 1, 'abono': 0, 'valor': 300000} {2541: {1: 300000}}
2	2541	1	25000.25487	{2541: {1: 300000}}	{'cliente': 2541, 'factura': 1, 'abono': 25000.25487, 'valor': 274999.74513} {2541: {1: 274999.74513}}
1	2541	2	500000	{2541: {1: 274999.74513}}	{'cliente': 2541, 'factura': 2, 'abono': 0, 'valor': 500000} {2541: {1: 274999.74513, 2: 500000}}
2	1429	5	25000.25487	{2541: {1: 274999.74513, 2: 500000}}	{'1429': 'No existe el cliente'} {2541: {1: 274999.74513, 2: 500000}}
1	1429	1	700000	{2541: {1: 274999.74513, 2: 500000}}	{'1429': 'No existe el cliente'} {2541: {1: 274999.74513, 2: 500000}}
2	1429	1	700000	{2541: {1: 274999.74513, 2: 500000}}	{'1429': 'No existe el cliente'} {2541: {1: 274999.74513, 2: 500000}}
0	1429	1	700000	{2541: {1: 274999.74513, 2: 500000}}	{'1429': 'No existe el cliente'} {2541: {1: 274999.74513, 2: 500000}}
0	1429			{2541: {1: 274999.74513, 2: 500000}}	{'1429': 'Cliente creado'} {2541: {1: 274999.74513, 2: 500000}, 1429: {}}
1	1429	1	700000	{2541: {1: 274999.74513, 2: 500000}, 1429: {}}	{'cliente': 1429, 'factura': 1, 'abono': 0, 'valor': 700000} {2541: {1: 274999.74513, 2: 500000}, 1429: {1: 700000}}
2	2541	1	274999.74513	{2541: {1: 274999.74513, 2: 500000}, 1429: {1: 700000}}	{'cliente': 2541, 'factura': 1, 'abono': 274999.74513, 'valor': 0.0} {2541: {2: 500000}, 1429: {1: 700000}}
3				{2541: {2: 500000}, 1429: {1: 700000}}	{'print': 'estado de la base de datos'} {2541: {2: 500000}, 1429: {1: 700000}}