Algoritmos y Estructuras de Datos
Trabajo Práctico 2 - 2025
Implementación de un Sistema Simple de Gestión de Órdenes de Pago (v2.0).

a. Introducción

Habiendo ya desarrollado el programa de Gestión de Órdenes de Pago para procesar una orden, ahora se desea continuar el desarrollo de manera que se puedan emitir informes respecto al procesamiento de **n** órdenes (esta cantidad representa un lote con todas las órdenes procesadas durante un día).

La empresa se maneja con las mismas cinco monedas válidas que se informaron para el TP1 (consultar el enunciado del TP1 para refrescar esa información).

A diferencia de la primera versión (TP1), donde se ingresaban los datos de una única orden de pago y por teclado, en esta segunda versión (TP2) los datos de las n órdenes vendrán dados en un archivo de texto llamado *ordenes.txt* y el programa que tienen que realizar no debe tomar datos desde el teclado por ninguna razón, ni debe tener un menú de opciones (eso implicaría cargar por teclado la opción elegida). Tampoco debe haber en ninguna parte nada que solicite al usuario presionar una tecla para continuar. Todos los datos que el programa debe procesar (y solo esos datos) estarán en el archivo de texto *ordenes.txt* cuya estructura se define más abajo.

Es MUY IMPORTANTE que ese archivo se llame *ordenes.txt* en el programa que desarrollen y entreguen, y que esté almacenado en la misma carpeta del proyecto que contiene al programa fuente. Si están haciendo pruebas con otros archivos, no hay problemas, pero ASEGÚRENSE de que cuando entreguen el programa para su calificación, la función *open()* esté abriendo exclusivamente un archivo llamado *ordenes.txt* Y NO OTRO.

El archivo de texto con *ordenes.txt* consistirá de una primera línea (que aparecerá por única vez) en la que de alguna forma no estándar se indicará la fecha y la hora de creación de ese archivo (una línea de este tipo en un documento, conteniendo fecha y hora de algún evento, se suele designar en el mundo informático como una *línea de marca temporal* (o *timestamp*)). En esa línea aparecerán (en cualquier orden y formato) los siguientes datos:

- Número del lote de datos representado por ese archivo.
- Fecha y hora de creación del lote de datos.
- Ciudad o barrio de la sucursal a la que pertenece ese lote de datos.

Algunos ejemplos (pero no los únicos posibles) de líneas de *timestamp* para este trabajo son los siguientes:

- 24563A3 - 22 de mayo de 2025 - 16:33 - Villa Cabrera
- 18hs 15ms - 15/10/2022 - Cofico - 993823
- Nueva Cordoba - 2024/03/27 - 9928371 - 20 horas

El resto de las líneas del archivo de entrada contendrá, cada una, los datos de una orden de pago, todos juntos conformando una cadena de 54 caracteres de largo (distribuidos como se explica más abajo), siempre con el mismo formato. Cada línea se separa de la que sigue con salto de línea. Si se tienen registrados los datos de 20 órdenes, entonces el archivo tendrá la línea de *timestamp* y luego otras 20 líneas, una por cada orden. En cada línea aparecerán, estrictamente en este orden, los siguientes datos:

- ***Caracteres del 0 al 19 (20 caracteres)***: El *nombre del destinatario* (rellenado a la derecha con espacios en blanco si fuese necesario hasta llegar a 20 caracteres).

- **Caracteres del 20 al 29 (10 caracteres):** El *número o código de identificación del destinatario* (rellenado a la derecha con espacios en blanco si fuese necesario hasta llegar a 10 caracteres).
- **Caracteres del 30 al 39 (10 caracteres):** El *código de la orden de pago* (rellenado a la derecha con espacios en blanco si fuese necesario hasta llegar a 10 caracteres). Igual que en el TP1, es una cadena de caracteres que puede puede tener cualquier formato, pero que necesariamente incluirá (en cualquier lugar) tres caracteres (en mayúscula) que indican la moneda en la cual se debe realizar el pago al beneficiario. Las monedas posibles son las mismas cinco que las usadas en el TP1, y se identifican con códigos de acuerdo al estándar ISO 4217 que ya se citó en el TP1.
- **Caracteres del 40 al 49 (10 caracteres):** El *monto nominal a transferir* (rellenado a la derecha con espacios en blanco si fuese necesario hasta llegar a 10 caracteres). Es un número entero que indica la cantidad en efectivo que *se ha solicitado pagar* (en la moneda indicada), al beneficiario. Igual que en el TP1, este monto nominal no es necesariamente el importe que se pagará, porque se pueden aplicar descuentos de distintos tipos que deberán restarse del monto nominal.
- **Caracteres 50 y 51 (2 caracteres):** El identificador del *algoritmo de cálculo de comisión* para calcular el monto base. Es un número entero de no más de dos dígitos (el segundo será un blanco si el número es de solo un dígito) que indica cuál es el algoritmo con el que se calcula la comisión para esta orden pago, de acuerdo a la tabla de *Algoritmos por Comisión* que se indica más abajo. Por ejemplo, si el número fuera 3, habría que calcular la comisión de acuerdo a las reglas especificadas en el algoritmo 3 de la citada tabla. *TODOS los algoritmos de cálculo de comisión tienen la misma entrada: moneda, tipo de algoritmo de cálculo de comisión y monto nominal.*
- **Caracteres 52 y 53 (2 caracteres):** El identificador del *algoritmo de cálculo impositivo* para calcular el monto final. Es un número entero de no más de dos dígitos (el segundo será un blanco si el número es de solo un dígito) que indica cuál es el algoritmo con el que se calculan los impuestos para esta orden de pago, de acuerdo a la tabla de Algoritmos Impositivos que se indica más abajo. Por ejemplo, si el número fuera 1, habría que calcular los impuestos de acuerdo a las reglas especificadas en el algoritmo 1 de la tabla citada. *TODOS los algoritmos de cálculo impositivo tienen la misma entrada: monto base y tipo de algoritmo de cálculo de impuesto.*

En el siguiente modelo mostramos qué forma podría tener un pequeño archivo de texto como el indicado:

```
18hs 15ms - 15/10/2022 - Cofico - 993823
Pedro Salazar       A16352232 01-354ARS 1555800   4 17
Andrea Montesinos   234BA998  USD87756  20000     1823
Alejandro BatistellaXC363684811JPN83A99930000000001245
```

En el modelo anterior la primera línea de todas es la de *timestamp*.

Luego siguen otras 3 líneas (una por cada orden pago registrada en ese archivo):

1. La primera es una orden a nombre de Pedro Salazar, número de identificación A16352232, código de orden de pago 01-354ARS, monto nominal 1555800, número de algoritmo para comisiones 4 y número de algoritmo para impuestos 17. Note en esta línea (y en todas las que siguen) los espacios en blanco al final de cada dato que lo requiera, para completar los caracteres que corresponden al formato de cada uno. Está claro que como la orden de pago contiene las letras ARS, entonces la moneda para esa transacción es el peso argentino.
2. La segunda es una orden a nombre de Andrea Montesinos, número de identificación 234BA998, código de orden de pago USD87756, monto nominal 20000, número de algoritmo para comisiones 18 y número de algoritmo para impuestos 23. La orden de pago contiene la cadena USD, por lo que la moneda para esta transacción es el dolar estadounidense

3. La tercera es una orden a nombre de Alejandro Batistella, número de identificación XC36368481, código de orden de pago 1JPN83A999, monto nominal 3000000000, número de algoritmo para comisiones 12 y número de algoritmo para impuestos 45. La orden de pago contiene la cadena JPN, por lo que la moneda para esta transacción es el yen. En esta línea se remarca en amarillo cada dato de por medio para facilitar la identificación, ya que en esta línea cada dato tiene el máximo número de caracteres permitido.

Respecto a la validez de una orden de pago, hay dos controles a realizar:

1. Debe haber, en el código de la orden, un código correspondiente al ISO 4217 de una de las cinco monedas válidas para este TP. Si no existe ninguno de esos cinco códigos de moneda, o hay dos o más diferentes, entonces la orden es inválida por "Moneda incorrecta". Note que si el mismo código de moneda está repetido dos o más veces, la orden es válida y esa moneda es la que se toma.
2. El código de identificación del destinatario de la orden de pago, debe componerse únicamente por letras mayúsculas, dígitos o guiones (pueden ser solo mayúsculas, o solo números, o cualquier combinación de mayúsculas, números y guiones, *pero no solo guiones*). Si no se cumple esta regla, o aparece cualquier caracter de otro tipo en este identificador, implicará que la orden de pago es inválida por "Destinatario mal identificado".

Cabe destacar que estos controles deben hacerse en ese orden. Esto significa que, si una orden tiene ambos controles incorrectos, se considera que es incorrecta por no tener una moneda válida. Si solo uno de esos controles es incorrecto, la orden es incorrecta por ese motivo.

A continuación mostramos las tablas de Cálculos de Algoritmos para Comisiones y para Impuestos:

Tabla de Algoritmos para Cálculo de Comisiones (Cálculo del Monto Base)	
Número	*Detalle*
1	Moneda: ARS comision = 9% del monto_nominal. monto_base = monto_nominal - comisión.
2	Moneda: USD Si el monto_nominal es menor a 50000, entonces comision = 0. Si el monto_nominal es mayor o igual a 50000, pero menor a 80000, entonces comision = 5% del monto_nominal. Si el monto_nominal es mayor a 80000, entonces comisión = 7.8% del monto nominal. monto_base = monto_nominal - comision.
3	Moneda: EUR o GBP Se cobra siempre un monto fijo de 100 en esa moneda, por gastos (monto_fijo = 100). Si el monto_nominal es mayor a 25000 (en esas monedas), entonces comision = 6% del monto_nominal. monto_base = monto_nominal - (monto_fijo + comision).
4	Moneda: JPY Si el monto nominal es menor o igual a 100000, entonces comision = 500 en todos los casos.

	Si el monto_nominal es mayor a 100000, entonces comision = 1000 en todos los casos. monto_base = monto_nominal - comision.
5	Moneda: ARS Si el monto_nominal es menor a 500000, entonces comision = 0. Si el monto_nominal es mayor o igual a 500000, entonces comision = 7% del monto_nominal. Pero si la comisión fuese mayor a 50000, entonces comision = 50000. monto_base = monto_nominal - comision.

Tabla de Algoritmos para Cálculo Impositivo (Cálculo del Monto Final)	
Número	*Detalle*
1	Si el monto_base es menor o igual a 300000, entonces impuesto = 0. Si el monto_base es mayor a 300000, entonces: excedente = monto_base - 300000 impuesto = 25% del excedente monto_final = monto_base - impuesto.
2	Si el monto_base es menor a 50000, entonces impuesto = 50 en todos los casos. Si el monto_base es mayor o igual a 50000, entonces impuesto = 100 en todos los casos. monto_final = monto_base - impuesto.
3	impuesto = %3 del monto_base para todos los casos. monto_final = monto_base - impuesto.

b. Requerimientos

Se pide desarrollar un programa que en base a todo lo anterior, procese los datos de un archivo con el formato indicado (la línea de *timestamp* debe ser ignorada), usando un único ciclo para levantar una por una las líneas del archivo, y muestre finalmente los siguientes resultados (estrictamente estos resultados, estrictamente en el orden que se indica):

1. La cantidad de operaciones inválidas que hay en el lote por moneda no autorizada (r1) y la cantidad de operaciones inválidas que hay en el lote por un destinatario mal identificado (r2). Recuerde: si una orden es inválida por ambos motivos, debe contarse como moneda no autorizada.

2. La cantidad de operaciones válidas (r3), y la suma de todos los montos finales de estas operaciones válidas (r4).

3. La cantidad de operaciones para cada una de las cinco monedas válidas en este modelo de trabajo (r5, r6, r7, r8 y r9). Si la orden tiene moneda válida, debe contarse sin importar si el beneficiario era incorrecto.

4. El código de la orden de pago (r10), el monto nominal (r11) y el monto final (r12) de la operación que tenga la mayor diferencia entre el monto nominal y el monto final (monto_nominal - monto_final) (si hubiera más de una operación con la misma diferencia mayor, informar el código de la orden de la primera de ellas). Aquí deben ser consideradas TODAS las órdenes de pago (tanto válidas como ínválidas…)

5. El nombre del beneficiario de la primera operación del archivo (r13), y la cantidad de veces que ese mismo beneficiario apareció en el archivo (r14).

6. El porcentaje que la cantidad de operaciones inválidas (por cualquier motivo) representa sobre la cantidad total de órdenes del archivo (r15). Para calcular este porcentaje, multiplique primero y luego haga la división, y para hacer la división, use el operador //.

7. El monto final promedio, pagado entre todas las órdenes válidas (en moneda y en beneficiario) emitidas en moneda ARS (r16). Para calcular este promedio, use el operador //.

c. Consideraciones especiales

El desarrollo de este TP2 debe realizarse exclusivamente en base a temas y prácticas presentados en las Fichas 1 a 13. El programa debe abrir el archivo de texto que se provea, procesar ese archivo línea por línea mediante un único ciclo, usar funciones parametrizadas y con retorno de valores en toda situación posible, y disponer de una función de entrada correctamente planteada. El uso de funciones en estas condiciones es obligatorio: el planteo del programa en base a un script gigante, sin funciones de ningún tipo o con funciones totalmente elementales, será motivo de aplazo inmediato.

En la Ficha 11 y en las clases prácticas de materia, se irán presentando temas adicionales necesarios para el desarrollo completo de este trabajo (incluyendo la forma de procesar línea por línea un archivo de texto). Mientras tanto, los estudiantes pueden ir investigando por sus propios medios la forma de hacer lo que necesiten hacer. Y por supuesto, siempre pueden preguntar en clases o por medio del foro para consultas sobre el TP2.

Repetimos para evitar sorpresas después: El uso de funciones correctamente parametrizadas y con retorno de valores si fuese aplicable, es obligatorio: el planteo del programa en base a un script gigante, sin funciones de ningún tipo o con funciones totalmente elementales, será motivo de aplazo inmediato.

Junto con este enunciado, se provee un archivo de prueba llamado *ordenes25.txt* con la línea de timestamp y luego 25 líneas con registros de prueba, para que los estudiantes puedan testear sus programas y validar básicamente lo que están haciendo. Para ese archivo en particular, los resultados que deberían obtener son los que se muestran en la captura de pantalla de la página siguiente.

Al igual que el TP1, este trabajo también será calificado en forma semiautomática (pero se controlará el cumplimiento del uso de funciones y el esquema general exigido): Tanto las entradas (que vendrán desde un archivo de texto) como las salidas, deben atenerse estrictamente a lo esperado.

El archivo de texto de entrada debe ser leído asumiendo el orden correcto de datos que contiene, y también es totalmente exigible el orden de las salidas, tal cual como se muestran en la captura para el archivo *ordenes25.txt* de la página siguiente.

```
 (r1) - Cantidad de ordenes invalidas - moneda no autorizada: 2
 (r2) - Cantidad de ordenes invalidas - beneficiario mal identificado: 12
 (r3) - Cantidad de operaciones validas: 11
 (r4) - Suma de montos finales de operaciones validas: 5091345568.0
 (r5) - Cantidad de ordenes para moneda ARS: 4
 (r6) - Cantidad de ordenes para moneda USD: 4
 (r7) - Cantidad de ordenes para moneda EUR: 6
 (r8) - Cantidad de ordenes para moneda GBP: 5
 (r9) - Cantidad de ordenes para moneda JPY: 4
(r10) - Codigo de la orden de pago con mayor diferencia nominal - final: MEURMUCGC
(r11) - Monto monto_nominal de esa misma orden: 835807747
(r12) - Monto final de esa misma orden: 589319388
(r13) - Nombre del primer beneficiario del archivo: Maria Robles
(r14) - Cantidad de veces que apareció ese mismo nombre: 4
(r15) - Porcentaje de operaciones inválidas sobre el total: 56
(r16) - Monto final promedio de las ordenes validas en moneda ARS: 534440912
```

No muestren nada más **en ninguna otra parte del programa**: Los únicos print() que el programa debe tener son los que están indicados en el enunciado para generar la secuencia de salidas pedidas. Y **nada más** que esos print() que se ven en la secuencia más abajo en esta misma página. No pongan títulos. No pongan mensajes de bienvenida. Ni saludos. Ni agradecimientos. No cambien caracteres en los mensajes de salida que indicamos más abajo. No agreguen líneas en blanco. No agreguen ningún print() de control. Nada. Limítense ESTRICTAMENTE a los print() que se indican más abajo.

Mostramos ahora la secuencia de instrucciones *print()* que sus programas deberían tener al final (copien y peguen al final de su código fuente toda la secuencia de salidas que sigue), y respeten a rajatabla **el orden de esas salidas, el formato, los mensajes y la cantidad de líneas** que les estamos mostrando. En la secuencia de abajo, lo único que pueden cambiar son los nombres de las variables en los que se suponen almacenados los resultados (obviamente, con esos nombres cada programador decide):

```
print(' (r1) - Cantidad de ordenes invalidas - moneda no autorizada:', cant_minvalida)
print(' (r2) - Cantidad de ordenes invalidas - beneficiario mal identificado:', cant_binvalido)
print(' (r3) - Cantidad de operaciones validas:', cant_oper_validas)
print(' (r4) - Suma de montos finales de operaciones validas:', suma_mf_validas)
print(' (r5) - Cantidad de ordenes para moneda ARS:', cant_ARS)
print(' (r6) - Cantidad de ordenes para moneda USD:', cant_USD)
print(' (r7) - Cantidad de ordenes para moneda EUR:', cant_EUR)
print(' (r8) - Cantidad de ordenes para moneda GBP:', cant_GBP)
print(' (r9) - Cantidad de ordenes para moneda JPN:', cant_JPY)
print('(r10) - Codigo de la orden de pago con mayor diferencia  nominal - final:', cod_my)
print('(r11) - Monto nominal de esa misma orden:', mont_nom_my)
print('(r12) - Monto final de esa misma orden:', mont_fin_my)
print('(r13) - Nombre del primer beneficiario del archivo:', nom_primer_benef)
print('(r14) - Cantidad de veces que apareció ese mismo nombre:', cant_nom_primer_benef)
print('(r15) - Porcentaje de operaciones inválidas sobre el total:', porcentaje)
print('(r16) - Monto final promedio de las ordenes validas en moneda ARS:', promedio)
```