Teste de Programação (Middle Office)

Instruções ao Candidato

Você deverá implementar um programa que simula a rotina de Middle Office da ARX, responsável por consolidar movimentações de cotistas e gerar boletas de aplicação e resgate.

Conceitos

- **FIC (Fundo de Investimento em Cotas)**: fundo aberto ao público (pessoas físicas, institucionais etc.). Os cotistas aplicam ou resgatam diretamente no FIC. O FIC **não possui ativos próprios**, ele recebe o financeiro de pessoas físicas e reinveste 100% do valor em um fundo **Master**.

- **Master**: fundo interno da ARX, não acessível diretamente ao público. É no Master que estão os ativos (ações, títulos, derivativos etc.). Cada FIC aplica ou resgata em um único Master, porém um Master pode ter vários FICs.

👉Fluxo: **Cotista → FIC → Master**

- **Custodiante**: banco que liquida (faz acontecer) as movimentações financeiras enviadas pelo Middle Office da ARX.
- Os FICs estão custodiados em um banco e os Masters em outro.
- Cada movimentação financeira entre fundos deve ter duas pontas de confirmação:
- **Ativo**: Informa qual fundo está enviando o dinheiro para qual fundo. É recebida pelo custodiante do fundo FIC.
- **Passivo**: Informa ao custodiante do Master que ele pode aceitar/liquidar essa movimentação.

Entrada de Dados

Os dados estarão em tabelas de banco de dados relacional (SQLite). O programa deve ler do banco, consolidar as informações e gerar **dois arquivos de saída** conforme layouts definidos.

Modelo de Dados

Tabela fundo

id	fundo_code	fundo_name	tipo
1	FIC_A	FIC A	FIC
2	FIC_B	FIC B	FIC
3	FIC_C	FIC C	FIC
4	Master_01	Master 01	Master
5	Master_02	Master 02	Master

Tabela fic_master_map

id	fic_id	master_id	comentário
1	1	4	FIC_A → Master_01
2	2	4	FIC_B → Master_01
3	3	5	FIC_C → Master_02

Tabela movements

id_movimento	cotista	fic_id	tipo	valor	datahora_mov
1	Joao	1	Aplicacao	1000.000000	2025-08-29 11:00:00
2	Maria	1	Aplicacao	500.000000	2025-08-29 11:05:00
...	...	...	...	...	...
1000+	...	...	...	...	entre 11:00 e 15:00

Observação: o banco será fornecido pronto. Você **não precisa** criar/popular.

Lógica a Implementar

- Ler no banco **apenas os movimentos do dia de processamento**.
- **Consolidar por FIC e por Tipo** (Aplicacao/Resgate), sem compensar sinais.
- Associar cada FIC ao seu **Master** via `fic_master_map`.
- **Evitar duplicidades**: proponha o seu mecanismo de controle.
- As mensagens retornadas por `processar_movimentacoes` devem indicar claramente se ainda há movimentos a importar para o custodiante.

Na geração, entregar **dois arquivos**: Ativo e Passivo.

Arquivo de Saída 1 – Boleta Ativo

Formato **CSV** separado por `;`, com o layout abaixo:

```
Data;FIC;Master;Tipo;Valor
29/08/2025;FIC_A;Master_01;Aplicação;1500.00
29/08/2025;FIC_B;Master_01;Resgate;200.00
```

Data	FIC	Master	Tipo	Valor
29/08/2025	FIC_A	Master_01	Aplicação	1500.00
29/08/2025	FIC_B	Master_01	Resgate	200.00

Arquivo de Saída 2 – Boleta Passivo

Formato **TSV** (tab) com layout abaixo:

```
VALOR   DATE    MASTER  FIC MOVIMENTO
1500.000000 08/29/2025  Master_01    FIC_A   RecebeAplicação
200.000000  08/29/2025  Master_01    FIC_B   RecebeResgate
```

VALOR	DATE	MASTER	FIC	MOVIMENTO
1500.000000	08/29/2025	Master_01	FIC_A	RecebeAplicação
200.000000	08/29/2025	Master_01	FIC_B	RecebeResgate

Regras de Negócio e Requisitos

1. **Sem boletas duplicadas:** se rodar novamente sem novidades, nada novo é gerado. O mecanismo de controle fica a seu critério, mas precisa ser claro e auditável.
2. **Dois arquivos por execução de geração.**
3. **Sem netting entre tipos:** Aplicação e Resgate consolidados separadamente por FIC.
4. **Rastreabilidade:** logs claros de quantidades lidas, consolidadas, descartadas e arquivos gerados.
5. **Reconciliação no final do dia:** verificar se todas as boletas que deveriam ser geradas foram de fato geradas.

Funções a Implementar e Exemplo de Uso

Você deve implementar **duas funções principais**:

```python
def processar_movimentacoes(data_processo: str) -> list[str]:
    """
    Lê os movimentos do banco na data informada, consolida as boletas
    e retorna mensagens de resumo (sem gravar arquivos).
    Deve respeitar, se presente, a variável de ambiente HORA_CORTE
(HH:MM:SS),
    filtrando somente movimentos com datahora_mov <= HORA_CORTE.
    """


def gerar_arquivos_boleta(data_processo: str, caminho_saida: str) -> None:
    """
    Gera os dois arquivos de saída (Ativo e Passivo) a partir da data
informada
    e do eventual HORA_CORTE (se definido em ambiente), no caminho indicado.
    """
```

Exemplo de uso em `__main__`

```
if __name__ == "__main__":
    import os
    data = "2025-08-29"

    mensagens = processar_movimentacoes(data)
    for msg in mensagens:
        print(msg)

    gerar = input("Deseja gerar os arquivos de boleta? (s/n):
").strip().lower()
    if gerar == "s":
        gerar_arquivos_boleta(data, "C:/TesteMiddle2025/saidas")
```

Cenário de Simulação

O avaliador poderá executar o programa diversas vezes no mesmo dia, simulando janelas de processamento. A cada rodada, o sistema deve filtrar as movimentações com base na variável de ambiente `` (HH\:MM\:SS). O banco conterá movimentações entre **11h e 15h**.

- O candidato não deve assumir previamente quais horários serão usados pelo avaliador.
- A cada rodada, o avaliador decide se deseja ou não gerar arquivos.

Requisitos de Testabilidade

- O código não deve depender de estado externo além do banco e do `HORA_CORTE`.
- Reexecutar a mesma rodada não deve duplicar linhas nos arquivos.
- As mensagens retornadas por `processar_movimentacoes` devem conter, no mínimo:
- Data e `HORA_CORTE` em uso.
- Quantidade de movimentos lidos e consolidados por FIC e Tipo.
- Totais consolidados do corte (Ativo e Passivo devem bater).
- Alertas de pendência (ex.: FIC sem mapeamento).
- Mensagem clara se ainda restam movimentos a importar para o custodiante.

Critérios de Avaliação

1. **Corretude** das consolidações em cada rodada/corte.
2. **Qualidade** e clareza das mensagens retornadas/exibidas.
3. **Layout** dos arquivos gerados exatamente conforme especificações.
4. **Controle de não duplicidade:**
5. A solução escolhida para garantir a não duplicidade será avaliada quanto à consistência e clareza.
6. A forma específica (uso de tabelas auxiliares, flags, etc.) fica a critério do candidato — inclusive é incentivado a criar tabelas adicionais no banco disponibilizado.
7. **Design** simples e legível do código.