Desenvolva um módulo em Python designado por **epp3.py**. Neste módulo deverá definir as funções descritas abaixo. Considere que o valor das transações são sempre números reais e as quantidades são sempre números inteiros. Não deverá preocupar-se em efetuar a geração/tratamento de exceções quando tal não lhe seja explicitamente pedido.

**1. Função** **total\_venda** **[4 valores]**

Escreva uma função em Python, chamada **total\_venda**, que recebe como argumento um tuplo contendo a cadeia de caracteres *"****VENDA****"*, um real positivo correspondente ao preço de cada unidade vendida e um inteiropositivo correspondente à quantidade de unidades vendidas. A função deverá devolver um real positivo correspondente ao preço total da venda. Quando o preço de cada unidade é igual ou inferior a 0, a função deverá gerar uma exceção do tipo **ValueError**, com a mensagem "**erro: preço deve ser um real positivo**". Quando a quantidade de unidades vendidas for igual ou inferior a 0, a função deverá gerar uma exceção do tipo **ValueError**, com a mensagem "**erro: quantidade deve ser um inteiro positivo**".

Exemplos

**>>> total\_venda(("VENDA", 2.5, 4))  
10.0  
  
>>> total\_venda(("VENDA", 0.0, 4))  
...  
ValueError: erro: preço deve ser um real positivo  
  
>>> total\_venda(("VENDA", 2.5, 0))  
...  
ValueError: erro: quantidade deve ser um inteiro positivo**

**1. Função** **total\_compra** **[4 valores]**

Escreva uma função em Python, chamada **total\_compra**, que recebe como argumento um tuplo contendo a cadeia de caracteres *"****COMPRA****"*, um real positivo correspondente ao preço de cada unidade comprada e um inteiropositivo correspondente à quantidade de unidades compradas. A função deverá devolver um real **negativo** correspondente ao preço total da compra. Quando o valor de cada unidade é igual ou inferior a 0, a função deverá gerar uma exceção do tipo **ValueError**, com a mensagem "**erro: preço deve ser um real positivo**". Quando a quantidade de unidades compradas for igual ou inferior a 0, a função deverá gerar uma exceção do tipo **ValueError**, com a mensagem "**erro: quantidade deve ser um inteiro positivo**".

Exemplos

**>>> total\_compra(("COMPRA", 2.5, 4))  
-10.0  
  
>>> total\_compra(("COMPRA", 0.0, 4))  
...  
ValueError: erro: preço deve ser um real positivo  
  
>>> total\_compra(("COMPRA", 2.5, 0))  
...  
ValueError: erro: quantidade deve ser um inteiro positivo**

**3. Função total [3 valores]**

Escreva uma função em Python, chamada total, que recebe como argumento um tuplo, que poderá corresponder às representações de uma venda ou de uma compra, conforme indicadas nas questões anteriores. A função deverá devolver o preço total da transação. Na implementação desta função, deverá chamar as funções **total\_venda**e **total\_compra**, conforme necessário. Quando o primeiro elemento da representação não corresponde a nenhuma das cadeias de caracteres "**VENDA**" ou "**COMPRA**", a função deverá gerar uma exceção do tipo **ValueError**, com a mensagem "**erro: transação inválida**".

Exemplos

**>>> total(("VENDA", 2.5, 4))  
10.0  
  
>>> total(("COMPRA", 2.5, 4))  
-10.0  
  
>>> total(("COMPRA", 0.0, 4))  
...  
ValueError: erro: preço deve ser um real positivo  
  
>>> total(("VENDA", 2.5, 0))  
...  
ValueError: erro: quantidade deve ser um inteiro positivo  
  
>>> total(("TESTE", 2.5, 4))  
...  
ValueError: erro: transação inválida**

**4. Função filtra\_por\_total [4 valores]**

Escreva uma função em Python, chamada **filtra\_por\_total**, que recebe como argumento uma lista de tuplos e um número, em que cada elemento da lista irá corresponder às representações de uma venda ou de uma compra, conforme indicadas nas questões anteriores. A função deverá devolver uma nova lista contendo todas as transações que tenham preço absoluto total não superior ao valor do segundo argumento (número). Utilize a função **abs** embutida no Python para obter o valor absoluto de um número.

Exemplo

**>>> filtra\_por\_total([("VENDA", 2.5, 4), ("COMPRA", 2.5, 4), ("VENDA", 2.0, 2), ("COMPRA", 2.0, 1)], 4)  
[('VENDA', 2.0, 2), ('COMPRA', 2.0, 1)]**

**5. Função resumo\_transacoes [5 valores]**

Escreva uma função em Python, chamada **resumo\_transacoes**, que recebe um número real (**saldo inicial**) e uma lista de tuplos, em que cada tuplo corresponde às representações de uma venda ou de uma compra, conforme indicadas nas questões anteriores. A função deverá devolver um tuplo com três elementos:  
**(1)** O saldo final após aplicação das transações ao saldo inicial. Ou seja, se o saldo inicial for 5 e houver uma compra com preço total 3, o saldo final é 2. Se o saldo inicial for 4 e houver uma venda com preço total 4, o saldo final é 8.  
**(2)** O menor preço total entre todas as compras. Caso não exista nenhuma compra, deverá retornar a constante **inf** disponível no módulo **math**.  
**(3)** O maior preço total entre todas as vendas. Caso não exista nenhuma venda, deverá retornar 0.

Exemplos

**>>> resumo\_transacoes(10.0, [("VENDA", 2.5, 4), ("COMPRA", 2.5, 4), ("VENDA", 2.0, 2), ("COMPRA", 2.0, 1)])  
(12.0, -10.0, 10.0)  
>>> resumo\_transacoes(10.0, [("VENDA", 2.5, 4), ("COMPRA", 2.0, 20), ("VENDA", 2.0, 2), ("COMPRA", 2.0, 1)])  
(-18.0, -40.0, 10.0)  
>>> resumo\_transacoes(5.0, [("COMPRA", 2.0, 20), ("COMPRA", 2.0, 1)])  
(-37.0, -40.0, 0)  
>>> resumo\_transacoes(5.0, [("VENDA", 2.0, 20)])  
(45.0, inf, 40.0)  
>>> resumo\_transacoes(5.0, [])  
(5.0, inf, 0)**

Notas Importantes

O seu trabalho não deve conter nenhuma instrução **print** nem **input**. As suas funções devem devolver valores e não escrevê-los. Assim, não deve utilizar instruções **print**, mas sim instruções **return**.

Os valores apresentados nos exemplos são apenas exemplificativos: as suas funções devem aceitar quaisquer valores para os respetivos parâmetros, dentro do tipo e gama especificados.

O programa pedido deverá ser submetido num ficheiro com o nome **epp3.py**. Na primeira linha do ficheiro deverá constar, em comentário, o seu nome e número de aluno.

Critérios de Avaliação

Cada uma das funções será classificada individualmente, para a cotação indicada, de acordo com os seguintes critérios.

Resultados corretos: 70%

*Se a função consegue produzir o resultado esperado, incluíndo, caso seja pedido, se consegue produzir as exceções corretas para parâmetros de entrada inválidos.*

Qualidade do código: 30%

*Se o código da função está corretamente comentado (não em excesso), se está bem estruturado e se é de fácil compreensão.*