Listas – mais exercícios

1. Faça um programa que percorre uma lista com o seguinte formato: [['Brasil', 'Italia', [10, 9]], ['Brasil', 'Espanha', [5, 7]], ['Italia', 'Espanha', [7,8]]]. Essa lista indica o número de faltas que cada time fez em cada jogo. Na lista acima, no jogo entre Brasil e Itália, o Brasil fez 10 faltas e a Itália fez 9.
O programa deve imprimir na tela:

a) o total de faltas do campeonato
b) o time que fez mais faltas
c) o time que fez menos faltas

Para os itens (b) e (c), você deve construir primeiro uma nova lista composta por sublistas com 2 elementos: [nome do país, total de faltas]

2. Escreva uma função em Pyhton, chamada **excluiExtremidades**, obrigatoriamente recursiva, que receba como parâmetros uma lista, cujos elementos podem ser de quaisquer tipos de dado, incluindo sublistas com diversos de níveis de profundidade, e um número inteiro (**n**) maior do que zero. Essa função deve excluir os **n** primeiros e os **n** últimos elementos da lista e de todas as suas sublistas.

Escreva um programa no qual será definida uma lista (**ls**) com as mesmas características da lista descrita acima e, em seguida, será lido do teclado um número inteiro (**n**) maior do que zero. Esse programa deverá exibir no monitor o conteúdo de **ls** antes e após a chamada da função **excluiExtremidades**.

Exemplos

ls = [1, 2, 3,[4.1,[4.21, 4.22, 4.23, 4.24], 4.3], 5, 6, 7]
n = 1
ls após a chamada de excluiExtremidades: [2, 3, [[4.22, 4.23]], 5, 6]

ls = [1, 2, 3,[4.1,[4.21, 4.22, 4.23, 4.24], 4.3], 5, 6, 7]
n = 2
ls após a chamada de excluiExtremidades: [3, [], 5]

ls = [1, 2, 3,[4.1,[4.21, 4.22, 4.23, 4.24], 4.3], 5, 6, 7]
n = 3
ls após a chamada de excluiExtremidades: [[ ]]

ls = [1, 2, 3,[4.1,[4.21, 4.22, 4.23, 4.24], 4.3], 5, 6, 7]
n = 4
ls após a chamada de excluiExtremidades: [ ]

Observação: note, a partir dos exemplos, que se a quantidade de elementos da lista (ou de uma sublista) for menor ou igual a **2*n**, ela se tornará uma lista (ou sublista) vazia.

3. Uma lista contém os nomes e as médias finais de cada um dos alunos inscritos em cada uma das disciplinas oferecidas em determinado semestre.

Exemplo

```
mediasFinais =    [ ['INF1025', [ [ 'joão',9.0 ], [ 'maria',8.0 ], [ 'josé',4.9 ] ] ],
                    ['INF1026', [ [ 'joão',7.0 ], [ 'maria',4.1 ] ] ],
                    ['INF1007', [ ['josé',4.3 ] ] ]
                  ]
```

Escreva uma função em Pyhton, chamada **gerarSituacaoFinal**, que receba como parâmetro a lista **mediasFinais** e retorne uma nova lista (**lst**) contendo **n** elementos, em que **n** representa o número de alunos (sem repetições) existentes na lista **mediasFinais**. Cada elemento de **lst** tem de conter o nome de um aluno e uma sublista, possivelmente vazia, com as disciplinas nas quais esse aluno foi aprovado (média final maior ou igual a 5,0).

Caso a função **gerarSituacaoFinal** receba a lista **mediaFinais** como parâmetro, a seguinte lista (**lst**) deverá ser retornada:

lst = [['joão', ['INF1025','INF1026']], ['maria', ['INF1025']], ['josé', []]]

Escreva um programa em Python para testar a sua implementação da função **gerarSituacaoFinal**.

4. O seguinte quadro classifica um curso d'água em função da sua DBO (Demanda Bioquímica de Oxigênio):

DBO	Classificação
Menor que 1	Muito limpo
Maior ou igual a 1 e menor que 2	Limpo
Maior ou igual a 2 e menor que 3	Razoável
Maior ou igual a 3 e menor que 4	Ruim
Maior ou igual a 4	Péssimo

a) Escreva uma função chamada **`classificaDBO`** que receba o valor de DBO de um curso d'água (valor real), uma lista com as classificações (strings - na mesma ordem da tabela acima) e retorne a classificação do curso d'água, conforme a tabela recebida.

b) Escreva uma função chamada **`buscaMaior`** que receba uma lista com os valores de DBOs (na qual cada elemento pode ser um valor real ou uma sublista com valores reais) e retorne o maior valor existente na lista.

c) Escreva uma função chamada **`piorClassificacao`** que receba uma lista com os valores de DBOs (valores reais) de um curso d'água, medidos em diferentes dias, e retorne a classificação (string) relativa ao maior valor da lista, conforme a tabela acima. Se houver mais de uma medição em um mesmo dia, os valores estarão organizados em sublistas. Por exemplo, na lista [3.9, [2.7, 7.8, 2.3, 5.6], 1.0, [9.0, 2.0], 2.0] há 1 registro de medição para o 1º dia (3.9), 4 para o 2º dia (2.7, 7.8, 2.3 e 5.6), 1 para o 3º dia (1.0), 2 para o 4º dia (9.0 e 2.0) e 1 para o 5º dia (2.0). Esta função deve, **obrigatoriamente**, usar as funções descritas nos itens **a** e **b**.

d) Faça uma função chamada **`classificaRios`** que receba uma lista em que cada elemento é composto por dois itens. São eles:

1. Nome do curso d'água;
2. Sublista com valores de DBOs de vários dias.

Exemplo: ['Rio PARAIBA DO SUL', [3.9, [2.7, 7.8, 2.3, 5.6], 1.0, [9.0, 2.0], 2.0]]

Esta função deve retornar uma nova lista em que cada elemento é composto por dois itens. São eles:

1. Nome do curso d'água;
2. Classificação do curso d'água.

Exemplo: ['Rio PARAIBA DO SUL', 'Péssimo']

Para testar a sua função, escreva um bloco principal que chame as funções descritas acima e utilize a seguinte lista de valores de DBOs:

```
lDBO = [['Rio AMAZONAS', [3.9, [2.7,2.3, 2.3, 5.6],1.0,[2.0,2.0], 2.0]],
        ['Rio URUGUAI', [[1.9, 1.8], 1.8, 0.3, 1.6, [1.0, 2.0, 1.9, 2.0]]],
        ['Rio TIETE', [3.9, [2.7, 3.0, 3.8, 2.3], [1.6, 1.0],[1.0, 2.0], 2.5]],
        ['Rio GUANDU', [4.0, [2.7, 7.8, 2.3, 5.6], [1.0, 9.0], 2.0, 2.0]],
        ['Rio DA PRATA', [0.02, [0.1, 0.1, 0.1], 0.05, 0.09, 0.08]],
        ['Rio MURIAE', [1.8, 1, 0.8, 4.0, 3.0]],
        ['Rio NEGRO', [[1.9, 1.8], 1.8, 0.3, 1.6, [1.0, 1.8, 1.9, 1.9]]]]
```

Utilizando a lista **`lBDO`** acima, a saída correta de seu teste deverá ser:

```
[['Rio AMAZONAS', 'Péssimo'], ['Rio URUGUAI', 'Razoável'], ['Rio TIETE', 'Ruim'],
['Rio GUANDU', 'Péssimo'], ['Rio DA PRATA', 'Muito limpo'],
['Rio MURIAE', 'Péssimo'], ['Rio NEGRO', 'Limpo']]
```