Listas – mais exercícios

1. Faça um programa que pergunte 6 nomes ao usuário, inserindo-os em uma lista. A seguir, leia vários nomes (término nome ==”fim”) e para cada um, caso já exista na lista, transforme-o em maiúsculo, senão insira-o. A cada nome processado, a lista deve ser exibida.

2. 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:

1. o total de faltas do campeonato
2. o time que fez mais faltas
3. 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]

3. O acesso aos quatro depósitos de um laboratório de Química exige a digitação de uma senha, previamente escolhida e cadastrada pelo usuário. Um usuário com acesso a diferentes depósitos terá de cadastrar senhas específicas para cada um deles. As permissões de acesso são definidas em função da categoria do usuário e são exibidas na tabela a seguir:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Categoria | Depósitos | | | |
| 1 | 2 | 3 | 4 |
| Aluno | Sim | Não | Não | Não |
| Professor | Sim | Sim | Não | Não |
| Pesquisador | Sim | Sim | Não | Não |
| Coordenador | Sim | Sim | Sim | Sim |

Por exemplo, um pesquisador terá de cadastrar duas senhas, uma para entrar no depósito 1 e outra para entrar no depósito 2. Por outro lado, um aluno terá de cadastrar uma única senha, que será usada para entrar no depósito 1.

Preocupados com possíveis invasões, os responsáveis pelo laboratório desejam saber o nível de segurança das senhas cadastradas pelos usuários. Para tal, escreva um programa que, para cada usuário, leia a sua categoria, as suas senhas e exiba no monitor o nível de segurança de cada uma delas.

O seu programa deve, **obrigatoriamente**, usar as seguintes funções, escritas por você:

1. **classificaUmaSenha**: recebe uma senha (string) e retorna o seu nível de segurança (string), conforme os critérios e níveis abaixo:

* Critérios:
  + 1. Possuir pelo menos 1 caractere numérico;
    2. Possuir pelo menos 1 caractere alfabético maiúsculo;
    3. Possuir pelo menos 1 dos seguintes caracteres: ! $ @ # % ( )
* Nível de Segurança:
  + - **FORTE**, caso a senha respeite os critérios i, ii e iii;
    - **MÉDIO**, caso a senha respeite o critério i, independentemente dos critérios ii e iii;
    - **RAZOÁVEL**, caso a senha respeite o critério ii, independentemente dos critérios i e iii;
    - **FRACO**, caso a senha respeite APENAS o critério iii;
    - **MUITO FRACO**, caso a senha não respeite critério algum.

1. **classificaTodasSenhas**: recebe a quantidade de depósitos que o usuário tem acesso e, para cada depósito, lê a respectiva senha e exibe no monitor o seu nível de segurança. Esta função deve retornar a quantidade de senhas com nível de segurança inferior ao nível **MÉDIO**.

**Observação:** use, **obrigatoriamente**, a função **classificaUmaSenha** para determinar o nível de segurança de uma senha.

1. **determinaQtdDepositos**: recebe o nome de uma categoria (string) e retorna a quantidade de depósitos que os usuários dessa categoria têm acesso.

Escreva um bloco principal no qual são lidas as categorias de um número indeterminado de usuários (a leitura da string 'FIM' indica o término dos dados). Para cada usuário use as funções anteriormente definidas para ler e exibir no monitor o nível de segurança de cada uma das senhas pertencentes a esse usuário.

Terminada a leitura dos dados de todos os usuários, o programa deve exibir o percentual de senhas com nível de segurança inferior ao nível de segurança **MÉDIO**. Esse percentual deve ser relativo ao somatório do número de senhas que cada usuário possui.

Exemplo (total de 4 usuários):

1. Usuário W - categoria Pesquisador 🡪 número de senhas: 2
2. Usuário X - categoria Coordenador 🡪 número de senhas: 4
3. Usuário Y - categoria Pesquisador 🡪 número de senhas: 2
4. Usuário Z - categoria Aluno 🡪 número de senhas: 1

O percentual de senhas deve ser relativo a **9** (2+4+2+1).

4. Faça uma função que receba uma lista qualquer e retorne o número total de caracteres das strings desta lista. A lista pode conter elementos do tipo *int, float, str, list e/ou bool*.

*Atenção:* caractere é todo elemento que pertence a uma string.

Exemplo: para a lista abaixo, a função retornaria 30.

lista = ["ano", 2016, ["saúde",2,7.2,'a?'], 'mais 1 email: feliz!']

5. Uma escola irá promover uma excursão para os alunos formandos do ensino médio. A escolha do hotel irá depender das quantidades de meninos e meninas inscritos. A regra é a seguinte:

1. Maior quantidade de meninas: “Pink Hotel”
2. Maior quantidade de meninos: “Hotel do Sport”
3. Mesma quantidade de meninos e meninas: “Le Hotel”

Faça uma função que receba uma lista de alunos onde cada elemento é uma sublista com nome e sexo do aluno inscrito e exiba qual o hotel escolhido.

Exemplo: Lista de inscritos = [ ['ana','f'],['pedro','m'],['carla','f'],['carlos','m']]

1. meninas e 2 meninos 🡪 Le Hotel

6. 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 |

1. 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.
2. 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.
3. 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**.
4. 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']]