Algoritmos e Técnicas de Programação
Engenharia Biomédica (2º ano)
Teste (época normal)
7 de Dezembro de 2023 (17h00)

Dispõe de **2:00 horas** para realizar este teste.
Alunos com necessidades especiais terão o tempo de **3:00 horas**.

Questão 1 (3v = 1+1+1)

Especifique as seguintes estruturas de dados em **compreensão**. Caso não consiga, pode apresentar uma solução em código com uma penalização de 25%:

a) Dicionário formado pelos pares que associam a cada número inteiro entre 1 e 20 inclusive, o seu cubo:

```
# Resultado esperado: {1: 1, 2: 8, 3: 27, 4: 64, ...}
dic1 = [...]
```

b) Lista formada pelas palavras do texto iniciadas por maiúscula:

```
texto = """Vivia há já não poucos anos algures num concelho do Ribatejo
um pequeno lavrador e negociante de gado chamado Manuel Peres Vigário"""
lista = [...]
```

c) Lista formada pelos números inteiros comuns às duas listas apresentadas:

```
lista1 = [1,3,5,7,9,11,13,15,17,19]
lista2 = [0,3,6,9,12,15,18]
# Resultado esperado: [3,9,15]
lista = [...]
```

Questão 2 (6v = 1.5+1.5+1.5+1.5)

a) Série numérica: Defina uma função `serie` que recebe três parâmetros, `base`, `salto` e `nelems`, e retorna a série ou sucessão respetiva.

Invocada com: `serie(7, 4, 5)` retornaria a lista: `[7, 11, 15, 19, 23]`

```
def serie(base, salto, nelems):
    ...
    return ...
```

b) Remover duplicados: Defina uma função que recebe uma lista como argumento e devolve uma nova lista sem elementos repetidos.

```
def remDupLista(lista):
    ...
    return ...
```

c) Divisores: Calcula a lista de divisores de um número inteiro (se um número divide o número passado como argumento dando resto zero na divisão inteira, é considerado um divisor desse número):

```
def divisores(n):
    ...
    return ...
```

d) É Primo?: Especifique uma função que verifica se um número inteiro é primo (tem apenas como divisores a unidade e o próprio número), dando como resultado True ou False (podes usar a função da alínea anterior); a seguir utiliza essa função para calcular uma lista em compreensão dos números primos entre 1 e 1000.

```
def ePrimo(n):
    ...
    return ...

primos1000 = [...]
```

Questão 3 (6v = 0.5+0.5+0.5+0.5+1+1+1+1)

Considere que a informação sobre um stock de uma loja está armazenada numa lista de tuplos (com o nome do produto, o seu preço unitário, e a quantidade em stock desse produto) de acordo com as seguintes estruturas em comentário:

```
# Produto = (nomeProd, preco, quantidade)
# designacao = String
# preco = Float
# quantidade = Int
#
# Stock = [Produto]
#
# Instância exemplo de uma papelaria
meuStock = [("caneta", 2.3, 34), ("lápis", 1.6, 22), ("caderno A4", 3.5, 50)]
```

Assuma que um produto não ocorre mais do que uma vez na lista de stock.
Defina as seguintes funções de manipulação e consulta do stock:

a) `quantos`, que indica quantos produtos existem em stock:

```
def quantos(stock):
    ...
    return ...
```

b) `emStock`, que indica a quantidade de um dado produto existente em stock. Se o produto não existir a função deverá retornar 0:

```
def emStock(nomeProd, stock):
    ...
    return ...
```

c) `consulta`, que indica o preço de um dado produto existente em stock. Se o produto não existir a função deverá retornar 0:

```
def consulta(nomeProd, stock):
    ...
    return ...
```

d) `tabPrecos`, que coloca uma tabela de preços (2 colunas: nome do produto e preço) na consola de output.

```
def tabPrecos(stock):
    ...
    return ...
```

e) `valorTotal`, que calcula o valor total do stock (se vendêssemos o stock inteiro quanto realizaríamos em dinheiro).

```
def valorTotal(stock):
    ...
    return ...
```

f) `saldos`, que gera um novo stock em que todos os produtos com quantidade acima de um valor de threshold estão em saldos de 25% (o preço do produto diminui em 25%).

```
def saldos(stock, threshold)::
    ...
    return ...
```

g) `maisBaratos`, que indica o(s) produto(s) mais barato(s) (se houver mais do que um produto com o preço mais baixo o resultado será uma lista desses produtos, se houver apenas um produto com o preço mais baixo o resultado será uma lista apenas com esse produto).

```
def maisBaratos(stock):
    ...
    return ...
```

h) `maisCaros`, que constrói a lista dos produtos caros (i.e., acima de um dado preço fornecido como parâmetro).

```
def maisCaros(valor, stock):
    ...
    return ...
```

Questão 4 (5v = 2+2+1)

No seguimento da pergunta anterior, considere o seguinte modelo de uma lista de compras:

```
# ListaCompras = {"nomeProd1": quantidade1, "nomeProd2": quantidade2, ...}
listaExemplo = {"caneta": 10, "caderno": 20}
```

E desenvolva as seguintes alíneas:

a) `verifLista`, que verifica se a lista de compras pode ser satisfeita, devolve como resultado 1 se sim e 0 caso contrário. Assuma que na lista de compras não há produtos repetidos.

```
def verifLista(lcompras, stock):
    ...
    return ...
```

b) `falhas`, que devolve a lista de itens (estrutura igual à lista de compras) que não podem ser comprados por não haver Stock suficiente na loja.

```
def falhas(lcompras, stock):
    ...
    return ...
```

c) `custoTotal`, que calcula o custo total da lista de compras (nesta alínea considere que o stock pode satisfazer a lista de compras).

```
def custoTotal(lcompras, stock):
    ...
    return ...
```