

P2 Alg 2025-1

Nome e turma: _____

1) Escreva uma função `maior_abaixo_media(lista)` que receba uma lista de números inteiros e devolva o maior número da lista, que seja menor que a média dos elementos. Resolva o problema em duas etapas: a) Calcule a média dos valores e crie uma nova lista apenas com os elementos menores que a média. b) Retorne o maior valor dessa nova lista. Se não houver nenhum valor menor que a média, a função deve retornar `None`. Não use as funções `sort`, `max`, nem bibliotecas. Apenas para o cálculo da média você pode usar `sum` e `len`. Exemplos: `maior_abaixo_media([10, 20, 30, 40]) -> 20`, `maior_abaixo_media([5, 5, 5]) -> None`, `maior_abaixo_media([1, 100]) -> 1`, `maior_abaixo_media([4, -8, 15, -16, 23, 42]) -> 4`

2) Crie uma função `zigue_zague(s)` que, dada uma string `s`, transforme em **maiúsculas** as letras que estiverem em posições **ímpares**. Exemplos: `zigue_zague('abacate')` -> `'aBaCaTe'`, `zigue_zague('')` -> `''`

3) O código abaixo foi escrito para somar os valores de uma lista de preços, mas contém vários erros distintos (de sintaxe e lógica). **a)** Encontre todos os erros **b)** Reescreva o código corrigido.

```
def soma(lista):
    soma = 0.0
    for p in lista:
        total = total + p
    return total
preços = "19.90 35.00 12.50 9.90".split()
print = ("Total: R$" + soma(preços))
```

4) def `pares_finais(n)`. A função recebe um número inteiro positivo `n`. Ela deve retornar quantos dígitos pares aparecem seguidos no final do número. Exemplos: `pares_finais(245680) -> 3`, `pares_finais(123456) -> 1`, `pares_finais(13) -> 0`.

5) **Enade 2011**. No livro "O Homem que Calculava", de Malba Tahan, um personagem desejava ganhar os grãos de trigos que fossem distribuídos sobre um tabuleiro de xadrez do seguinte modo: um grão na primeira casa do tabuleiro, o dobro (2) na segunda, novamente o dobro (4) na terceira, outra vez o dobro (8) na quarta, e assim por diante, até a sexagésima quarta casa do tabuleiro. Faça um algoritmo que calcule a **quantidade total** de grãos de trigo necessários para realizar esta distribuição. **Não use o operador `**`** de exponenciação neste exercício.

6) Crie uma função `mesmas_letras(s1, s2)` que verifique se as strings `s1` e `s2` contêm exatamente as **mesmas letras, independentemente da ordem e da quantidade**. Exemplos: `mesmas_letras('banana', 'annaba') -> True`, `mesmas_letras('casa', 'casal') -> False`, `mesmas_letras('casal', 'casa') -> False`, `mesmas_letras('amor', 'roma') -> True`. Apesar de ser possível de ser

feito em uma única linha, você pode usar while ou for e fazer em várias linhas.

7) def maldição(s). A entrada é uma string `s`, com letras minúsculas e sem acentos, o retorno é uma nova string, onde a vogal que mais aparece deve ser trocada por '#'. Se houver empate pegue a primeira vogal mais frequente. Use `count()` para contar quantas vezes cada vogal aparece. Depois, use `replace()` para fazer a troca. Exemplos: `maldição('abracadabra bobo sem nocao')` -> '#br#c#d#br# bobo sem noc#o'
`maldição('aaaxxxeeexxiixxxooxxxxuuu')` -> '###xxxeeexxiixxxooxxxxuuu'

8)def sanduíche_com_vogais(s). Conta quantas vezes um padrão "sanduíche com vogais" aparece em uma string. Um sanduíche com vogais é uma sequência de 3 caracteres onde o primeiro e o terceiro são vogais e o segundo pode ser qualquer caractere. Exemplos: `sanduíche('amazing')` -> 2, `sanduíche('coool')` -> 2, `sanduíche('aeiouAEIOU')` -> 8, `sanduíche('')` -> 0

9)def soma_duplas(nums). Dada uma lista de inteiros `nums`, some cada número que aparece **exatamente duas vezes**, apenas uma vez na soma final. Exemplos: `soma_duplas([1, 2, 2, 3, 3, 3, 4, 4])` -> 2 + 4 = 6, `soma_duplas([1, 1, 2, 2, 3])` -> 1 + 2 = 3, `soma_duplas([])` -> 0.

10) Mostre a resposta do código abaixo:

```
x = ['', 'abacate', {}, {42: 'resposta'}, [], [42], 0, 42, 3 == 3.0, '' in 'abacate']
res = []
while x:
    item = x[0]
    x = x[1:]
    if item:
        res.append(item)
    else:
        res.append('Falso lógico')
print (res)
if [42] in res: print ('Eu sou Feliz!')
print (res.count('Falso lógico'))
print (res.count(True))
print (res.count('Falso'))
if False: print ('Falso!')
if True: print ('Vou aprender mais!')
```

11) Bônus. Implemente a função `espelho_frase(frase)` que recebe uma string contendo uma frase formada por várias palavras separadas por espaço e devolve uma nova string onde a ordem das palavras é invertida e cada palavra é transformada em seu espelho, ou seja, seus caracteres aparecem na ordem inversa. Não utilize estruturas de repetição como `for` ou `while`. Utilize apenas funções e métodos de string, como `split()`, `join()` e fatiamento com `[::-1]`. Exemplos: `espelho_frase("Python é Legal")` -> "lageL é nohtyP"