MAC2166 - Introdução a Computação - 2024S1 **Avaliação P3**

0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9

Utilize caneta azul ou preta e preencha completamente a quadrícula.
Exemplo: ■. **Não use ☒.**

Turma: (somente um número; consulte a pessoa responsável se não souber)

4	5	6	7	8	9	10	11	12	13	14	20

⟵ Marque as quadrículas ao lado para formar o seu número USP e escreva seu nome completo em letra legível na linha pontilhada abaixo. **Se seu número possui menos que 8 dígitos complete com zeros à esquerda.**

Nome:

.....

Esta prova tem duração de 120 minutos. Não desmonte a prova.

Q1 [1,5 pontos] Simule o código abaixo e selecione as opções correspondentes às linhas impressas pelo programa (**Nenhuma linha** deve ser selecionada caso o programa não imprima a enésima linha)

```
def f1(s):
    r = 10 - len(s)%10
    return (s + str(r) + str(r))
def f2(c):
    o = c
    if c == "1":
        o = "One"
    elif c == "3":
        o = "Three"
    elif c == "6":
        o = "Six"
    return o
def main():
    L = "aeiou"
    F = "MAC2166 "
    K = L.upper()
    print(len(F))
    G=f1(F[:7])
    for i in range(len(G)-5):
        if G[i] not in L:
            print(f2(G[-i]))
    print(f2(G[i+1]), f2(G[i+2]), end="")
    print(G[8])
main()
```

Rascunho

Selecione a primeira linha:
☐ 1 ☐ 7 ☐ 8 ☐ 5 ☐ AEIOU

Selecione a segunda linha:
☐ M ☐ MAC216633 ☐ 3 ☐ MAC2166 33 ☐ 8

Selecione a terceira linha:
☐ M ☐ MAC216633 ☐ A ☐ Three ☐ MAC2166 33

Selecione a quarta linha:
☐ M ☐ A ☐ 3 ☐ Three ☐ C

Selecione a quinta linha:
☐ A ☐ C ☐ Three ☐ Six ☐ 2

Selecione a sexta linha:
☐ Two ☐ One ☐ C ☐ Three ☐ One Six3

Selecione a sétima linha:
☐ Nenhuma linha ☐ One ☐ Six ☐ Three ☐ One Six3

Selecione a oitava linha:
☐ 3 ☐ Nenhuma linha ☐ One ☐ Six ☐ One Six3

Q2 [2,5 pontos]

Nesta questão há dois arquivos, um arquivo de entrada e um arquivo de saída. O arquivo de entrada possui os dados das notas dos alunos de uma turma em um curso especial e está organizado em quatro linhas nas quais os dados são separados por vírgula. A linha 0 contém os nomes dos alunos de uma turma; a linha 1 contém o Número USP dos alunos desta turma; a linha 2 contém as notas da primeira prova do curso; e a linha 3 contém as notas da segunda prova do curso. Esses dados estão organizados de forma que cada coluna corresponde aos dados de um mesmo aluno.

O objetivo desta questão é fazer uma função que cria um arquivo transposto no qual substituímos as duas notas de cada aluno pela média usp, uma média fictícia incluída no módulo `pacote_usp` como uma função de nome `média_usp`, que recebe duas notas e retorna um novo valor. Queremos escrever uma função que recebe duas strings com os nomes dos arquivos de entrada e saída, respectivamente, e escreve um arquivo (apagando o conteúdo previamente existente nesse arquivo) no qual cada linha corresponde a um aluno. Por exemplo, se a média for calculada por $(nota_1 + 2 * nota_2)/3$ e o arquivo de entrada for

Longina Diana, Nóra Pradeep, Juancho Ramazan
67363, 43392, 85114
2, 5, 8
3, 7, 7

O arquivo de saída deve ser

Longina Diana, 67363, 2.6666666666666665
Nóra Pradeep, 43392, 6.333333333333333
Juancho Ramazan, 85114, 7.333333333333333

```
import pacote_usp

def transposta_usp(entrada, saida):
    input_file = open(entrada, L1 )
    output_file = open(saida,L2 )

    dados = [ ]
    for linha in input_file:
        dados.append(linha.rstrip().split(L3 ))
    input_file.close()

    for i in range(len(dados[0])):
        output_file.write(L4 .strip() + ", ")
        output_file.write(L5 .strip() + ", ")
        nota1, nota2 = int(dados[2][i]),L6
        media = L7 (nota1,nota2)
        output_file.write(L8 )
    output_file.close()
```

Preencha as lacunas no código acima (L1 até L8), assinalando as respostas correspondentes abaixo.

Consideração: Para cada lacuna, assinale no máximo uma resposta.

L1:	☐ read	☐ r	☐ a	☐ 'r'	☐ 'w'	
L2:	☐ write	☐ append	☐ 'w'	☐ 'a'	☐ r	
L3:	☐ ;	☐ ","	☐ "\n"	☐ " "	☐ ,	
L4:	☐ dados[0]	☐ dados[0][0]	☐ dados[1][i]	☐ dados[i][0]	☐ dados[0][i]	
L5:	☐ dados[1][i]	☐ dados[1]	☐ dados[0][i]	☐ dados[1][0]	☐ dados[i][0]	
L6:	☐ int(dados[3][i])	☐ str(dados[3][i])	☐ nota1	☐ int(dados[2][i])	☐ dados[3][i]	
L7:	☐ pacote_usp	☐ média_usp	☐ pacote_usp.media	☐ média_usp.pacote_usp	☐ pacote_usp.média_usp	
L8:	☐ str(media)	☐ str(media), end = "\n"	☐ int(media) + "\n"	☐ media + "\n"	☐ str(media) + "\n"	

Q3 [2,5 pontos]

Você foi convidado a integrar uma equipe responsável por desenvolver um pequeno sistema para gerenciar o empréstimo de livros do Grêmio da POLI. Sua tarefa foi escrever uma bateria de testes automatizados para o sistema que possui as seguintes funções:

```
def insere_livro(titulo, autor, ano):
   # devolve um código único para o livro inserido e o registra como disponível
def busca_livro(titulo):
   # devolve o código de um livro que possui o titulo convidado
   # ou devolve -1 caso o livro não tenha sido encontrado
def empresta_livro(codigo):
   # devolve True caso o livro esteja disponível e o registra como emprestado
   # devolve False caso o livro esteja já emprestado
def devolve_livro(codigo):
   # devolve True caso o livro esteja emprestado e o registra como disponível
   # devolve False caso o livro não esteja emprestado ou não exista
def limpa():
   # remove todos os livros da biblioteca
```

```
import pytest
import biblioteca
def monta_biblioteca():
    biblioteca.limpa()
    biblioteca.insere_livro("Amor nos tempos do cólera", "Gabriel Garcia Marques", 1984)
    biblioteca.insere_livro("Engineering for Dummies", "John Eng", 1922)
    biblioteca.insere_livro("Manual do politécnico revolucionário", "Carlos Marcos", 1867)

def testa_emprestimo_com_sucesso():
    monta_biblioteca()
    cod1 = biblioteca.busca_livro("Amor nos tempos do cólera")
    L1
    r1, r2 = biblioteca.empresta_livro(cod1), biblioteca.empresta_livro(cod2)
    L2

def testa_emprestimo_repetido():
    monta_biblioteca()
    cod1 = biblioteca.busca_livro("Amor nos tempos do cólera")
    L3
    assert (r1, r2) == (True, False)

def testa_devolucao_com_sucesso():
    monta_biblioteca()
    cod1 = biblioteca.busca_livro("Amor nos tempos do cólera")
    L4
    L5
    assert (r1, r2) == (True, False)

def testa_devolucao_sem_emprestimo():
    monta_biblioteca()
    L6
```

Preencha as lacunas no código acima (L1 até L6), assinalando as respostas correspondentes abaixo.

Consideração: Para cada lacuna, assinale no máximo uma resposta.

L1:

- ☐ `cod1 = biblioteca.busca_livro("Engineering for Dummies")`
- ☐ `cod1 = biblioteca.busca_livro("Carlos Marcos")`
- ☐ `cod3 = biblioteca.busca_livro("Engineering for Dummies")`
- ☐ `cod2 = biblioteca.busca_livro("Amor nos tempos do cólera")`
- ☐ `cod2 = biblioteca.busca_livro("Engineering for Dummies")`

L2:
- [ ] `assert (r1, r2) == (True, False)`
- [ ] `assert (r1, r2) == (False, True)`
- [ ] `assert (r1, r1) == (False, False)`
- [ ] `assert (r1, r2) == (True, True)`
- [ ] `assert (r1, r2) == (False, False)`

L3:
- [ ] `r1, r2 = biblioteca.empresta_livro(cod1), biblioteca.empresta_livro(cod2)`
- [ ] `r1, r1 = biblioteca.empresta_livro(cod1), biblioteca.empresta_livro(cod1)`
- [ ] `cod1, cod2 = biblioteca.empresta_livro(r1), biblioteca.empresta_livro(r2)`
- [ ] `cod2, cod1 = biblioteca.empresta_livro(r2), biblioteca.empresta_livro(r1)`
- [ ] `r1, r2 = biblioteca.empresta_livro(cod1), biblioteca.empresta_livro(cod1)`

L4:
- [ ] `biblioteca.devolve_livro(cod1)`
- [ ] `biblioteca.empresta_livro(cod)`
- [ ] `devolve_livro(cod1)`
- [ ] `empresta_livro(cod1)`
- [ ] `biblioteca.empresta_livro(cod1)`

L5:
- [ ] `r1, r2 = biblioteca.devolve_livro(cod1), biblioteca.devolve_livro(cod2)`
- [ ] `r1, r2 = biblioteca.devolve_livro(cod1), biblioteca.devolve_livro(cod1)`
- [ ] `r1, r2 = biblioteca.empresta_livro(cod1), biblioteca.empresta_livro(cod1)`
- [ ] `r2, r1 = devolve_livro(cod1), devolve_livro(cod1)`
- [ ] `r1, r2 = biblioteca.empresta_livro(cod1), biblioteca.devolve_livro(cod1)`

L6:
- [ ] assert biblioteca.devolve_livro(cod1) == False
- [ ] assert devolve_livro(0) == True
- [ ] assert biblioteca.devolve_livro(0) == True
- [ ] assert devolve_livro(0) == False
- [ ] assert biblioteca.devolve_livro(0) == False

Q4 [2,5 pontos] O Bucket sort é um algoritmo de ordenação que funciona da seguinte forma. Suponha que recebemos uma lista *Lista* com n números não negativos, cujo maior elemento é M. Então criamos n "baldes" (buckets), e gostaríamos de dividir os elementos de *Lista* de forma que o j-ésimo balde tenha os números de *Lista* que estão no intervalo $[j * \frac{M}{n}, (j+1) * \frac{M}{n})$ (para $j = 0, 1, \ldots, n-2$), e que o $(n-1)$-ésimo balde tenha os números de *Lista* que estão no intervalo $[(n-1) * \frac{M}{n}, M]$. Desta forma, se x está em *Lista*, então x é colocado precisamente no $\lfloor x * n/M \rfloor$-ésimo balde se $x \neq M$, e se $x = M$, então x é colocado no $(n-1)$-ésimo balde.

Por exemplo, considere $Lista = [5, 85, 56, 91, 92, 18, 54, 75, 94, 12]$ que possui dez elementos. A lista de buckets criada é

$$\big[[5], [18, 12], [\,], [\,], [\,], [56, 54], [\,], [75], [\,], [85, 91, 92, 94]\big].$$

Finalmente, nós ordenamos cada balde individualmente usando a função `ordena`, que ordena uma sequência de números, concatenamos os baldes ordenados, e retornamos a lista obtida.

```
def bucket_sort(Lista):
    n, máximo = len(Lista), max(Lista) # max() é uma função do python que devolve o valor máximo de uma lista.
    comp = máximo/[L1]

    lista_de_buckets= []
    for j in range(n):
        [L2].append([])

    for x in [L3]:
        j = int(x / comp)
        if j < n:
            [L4].append(x)
        else:
            lista_de_buckets[n - 1].append([L5])

    for j in range(n):
        ordena([L6])

    saída_final = [L7]
    for bucket in [L8]:
        for x in bucket:
            saída_final.append(x)

    return saída_final
```

Preencha as lacunas no código acima (L1 até L8), de forma a obter a função descrita.

OBS: Para cada lacuna, assinale no máximo uma resposta.

L1:	☐ 0	☐ n	☐ 2	☐ 2*n	☐ 1	
L2:	☐ comp	☐ range(n)	☐ lista_de_buckets	☐ Lista	☐ x	
L3:	☐ range(n)	☐ lista_de_buckets	☐ Lista	☐ n	☐ [ ]	
L4:	☐ lista_de_buckets	☐ range(n)	☐ Lista	☐ x	☐ lista_de_buckets[j]	
L5:	☐ comp	☐ j	☐ x	☐ máximo	☐ n	
L6:	☐ Lista	☐ range(n)	☐ lista_de_buckets[j]	☐ lista_de_buckets	☐ lista_de_buckets[n]	
L7:	☐ Lista	☐ lista_de_buckets	☐ range(n)	☐ n	☐ [ ]	
L8:	☐ range(máximo)	☐ saída_final	☐ lista_de_buckets	☐ Lista	☐ range(n)	

Q5 [1 ponto] Sobre (1) Módulos em Python e (2) Testes Automatizados, podemos afirmar, respectivamente que:

- ☐ (1) são uma forma de tornar o código menor, potencialmente evitando-se a escrita de muitas linhas de código; (2) devem ser evitados pois testes manuais executados por especialistas no assunto são mais efetivos do que testes escritos por programadores que podem ser inexperientes.
- ☐ (1) permitem agrupar uma ou mais funções dentro de um arquivo (módulo) que pode ser acessado a partir de outros arquivos (módulos); (2) quando bem escritos, oferecem a certeza de que o código está correto.
- ☐ (1) permitem agrupar uma ou mais funções dentro de um arquivo (módulo) que pode ser acessado a partir de outros arquivos (módulos); (2) quando bem escritos, oferecem aos desenvolvedores uma maior segurança de que o código provavelmente está correto, facilitando o processo de desenvolvimento de programas complexos.
- ☐ (1) permitem agrupar uma ou mais funções dentro de um arquivo (módulo) que pode ser acessado a partir de outros arquivos (módulos); (2) devem ser evitados pois testes manuais executados por especialistas no assunto são mais efetivos do que testes escritos por programadores que podem ser inexperientes.
- ☐ (1) são uma forma de aumentar a eficiência do código, tornando-o mais rápido; (2) quando bem escritos, oferecem aos desenvolvedores uma maior segurança de que o código provavelmente está correto, facilitando o processo de desenvolvimento de programas complexos.
- ☐ (1) permitem agrupar funções em diferentes arquivos desde que as funções não possuam o mesmo nome; (2) devem ser executados manualmente pelos programadores.

Com base nas alternativas acima, assinale a única alternativa correta.