

MC102  
Algoritmos e Programação de Computadores  
Exame  
Turmas A B C D E F I J K L  
Segundo Semestre de 2019

Questão	Nota
1.a	
1.b	
1.c	
2	
3	
4	
Total	

Nome: RA: 

**Importante:** Esta avaliação é individual. Não é permitida a consulta a qualquer material impresso, nem o uso de celulares ou outros dispositivos eletrônicos de comunicação/computação. As respostas deverão ser escritas nos espaços demarcados, opcionalmente a lápis. Não se esqueça de indentar corretamente os códigos solicitados. **Boa prova!**

1. Em cada série, para cada trecho de código, indique o que será escrito quando os programas forem executados. Caso um programa execute corretamente e não produza nenhuma saída, preencha com “—” o espaço da resposta correspondente. Caso algum erro seja encontrado, indique o motivo e marque no código o ponto em que ele ocorre.

a) (1.2 ponto) **Listas, tuplas e dicionários**

```
lista = [1, 2, 3]
lista.append(4.0)
soma = 0
for i in range(len(lista)):
    soma = soma + lista[i]
print(soma)
```

```
lista = [3, 6]
lista = lista + [9]
print(lista)
lista = lista + "12.0"
print(lista)
```

```
lista = [("A", 1), ("B", 2)]
lista[0] = ("A", "a")
print(lista)
lista[1] = ["B", "b"]
lista[2] = ("C", "c")
print(lista)
```

```
A = [[0], [1, 2], [1, 2, 3]]
for i in range(len(A)):
    for j in range(len(A[i]) - 1):
        A[i][j] = A[i][j] + i
print("A = ", A)
```

```

frequencia = {"C":750, "Pascal":100, "Java":300}
for linguagem in frequencia:
    if frequencia[linguagem] < 200:
        print(linguagem, ":", frequencia[linguagem])
frequencia["Python"] = 1200
print("Python :", frequencia["Python"]),
frequencia["Ada"] = frequencia["Ada"] + 1
print("C :", frequencia["C"])

```

```

Pascal : 100
Python : 1200
KeyError: 'Ada'

```

b) (0.9 ponto) **Funções, passagem de parâmetros e escopo de variáveis**

```

def tupla(a, b):
    return (a, b)

```

```

print(tupla(0, 5))

```

```

(0, 5)

```

```

def soma(a, b):
    return a + b + c

```

```

c = 5
print(soma(0, 5))

```

```

10

```

```

def tupla(a, b):
    t = (a, b)

```

```

tupla(10, 20)
print(t)

```

```

NameError: name 't' is not
defined

```

c) (0.9 ponto) **Recursão**

```

def recursiva(n) :
    if n == 1 :
        print("@")
    else:
        recursiva(n-1)
        print(n * "*")

```

```

recursiva(4)

```

```

@
**
***
****

```

```

def recursiva(n) :
    if n == 1 :
        print("@")
    else:
        recursiva(n-2)
        print(n * "*")
        recursiva(n-2)

```

```

recursiva(3)

```

```

@
***
@

```

```

def recursiva(n) :
    if n == 1 :
        print("@")
        recursiva(n-2)
        print(n * "*")
        recursiva(n)

```

```

recursiva(4)

```

```

RecursionError: maximum
recursion depth exceeded in
comparison

```

**Dica:** um número  $n$  multiplicado por uma string retorna a concatenação de  $n$  cópias desta string. Por exemplo,  $5 * "*"$  retorna  $"*****"$ .

2. (2.0 pontos) Observe as matrizes abaixo de maneira a identificar um padrão.

$$\begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

O elemento  $m[0][0]$  da matriz abaixo, implementada em Python por uma lista de listas, não segue o padrão acima. Indique, no formato  $m[i][j]$ , pelo menos mais três elementos que também não seguem o padrão.

```
m = [[1, 1, 0, 0, 0, 5],  
      [0, 1, 0, 0, 0, 5],  
      [0, 1, 0, 0, 0, 5],  
      [0, 1, 0, 0, 0, 5],  
      [0, 1, 0, 0, 0, 5],  
      [0, 1, 0, 0, 0, 5]]
```

$m[0][5]$   $m[1][5]$   $m[2][5]$   $m[3][5]$   $m[4][5]$   $m[5][5]$

Considerando este modelo de representação de matrizes, escreva uma função `verifica_padrao(m)` que retorna `True` se uma matriz `m` passada como parâmetro for quadrada, tiver altura maior do que 2 e respeitar o padrão ou `False` caso contrário. Utilize o comando `len()` para obter as dimensões de `m`.

```
def verifica_padrao(m) :  
    altura = len(m)  
    if altura < 2 :  
        return False  
    for i in range(altura):  
        largura = len(m[i])  
        if largura != altura :  
            return False  
        for j in range(largura):  
            if j != 1 and m[i][j] != 0 :  
                return False  
            elif j == 1 and m[i][j] != 1 :  
                return False  
    return True
```

3. (2.0 pontos) Susana começou a estudar algoritmos de ordenação e está explorando o comportamento da função abaixo. Para acompanhar os passos do algoritmo codificado, Susana introduziu algumas chamadas ao comando `print()` em pontos estratégicos: no início da função e após algumas movimentações dos elementos.

```
def ordena(lista):  
    print("lista =", lista)  
    for i in range(1, len(lista)):  
        aux = lista[i]  
        j = i - 1  
        while j >= 0 and aux < lista[j]:  
            lista[j+1] = lista[j]  
            j = j-1  
        lista[j+1] = aux  
    print("lista =", lista)
```

Como seu primeiro teste, Susana fez a seguinte chamada para a função:

```
ordena([7, 6, 5, 8, 4, 3, 2, 1])
```

Abaixo, está indicado o que será escrito pela primeira chamada ao comando `print()`. Seguindo o modelo, complete os espaços com o que será escrito pelas próximas cinco chamadas:

lista = [ 7, 6, 5, 8, 4, 3, 2, 1 ]

lista = [ 6, 7, 5, 8, 4, 3, 2, 1 ]

lista = [ 5, 6, 7, 8, 4, 3, 2, 1 ]

lista = [ 5, 6, 7, 8, 4, 3, 2, 1 ]

lista = [ 4, 5, 6, 7, 8, 3, 2, 1 ]

lista = [ 3, 4, 5, 6, 7, 8, 2, 1 ]

Qual é o nome do algoritmo implementado?

Qual seria o número total de linhas necessárias para escrever o funcionamento completo da função `ordena()` com a entrada fornecida?

4. (3.0 pontos) As  $n$  tarefas de laboratório propostas em MC502 têm pesos que variam entre 1 e 5. De acordo com o critério de avaliação descrito no Plano de Desenvolvimento da Disciplina, um(a) aluno(a) precisa ter **média ponderada mínima** 6.0 e **nota mínima** 3.0 nas tarefas com peso maior ou igual a 3 para poder ser aprovado(a) sem exame. Escreva uma função

`desempenho_labs_suficiente(labs)`

que retorna `True` caso estes critérios tenham sido satisfeitos ou `False` caso contrário. Suponha que a função irá receber uma lista de tuplas `labs` no seguinte formato:

`labs = [ (nota0, peso0), (nota1, peso1), ..., (nota $n-1$ , peso $n-1$ ) ]`

```
def desempenho_labs_suficiente(labs) :
    peso_labs = 0
    nota_labs = 0
    for lab in labs :
        if lab[0] < 3.0 and lab[1] <= 3:
            return False
        nota_labs += lab[0] * lab[1]
        peso_labs += lab[1]

    media = nota_labs / peso_labs

    return media >= 6.0
```