MC102

Algoritmos e Programação de Computadores

Prova 2

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

[3, 6, 9, '12.0']

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

TypeError: can only concatenate list (not "int") to list

```
lista = [3, 6]
lista[2] = 9
print(lista)
```

IndexError: list assignment index out of range

```
lista = [("A", 1), ("B", 2)]
lista[0] = ["C", 3]
lista[0][1] = 4
print(lista)
```

[['C', 4], ('B', 2)]

```
A = [[0, 1], [1, 2], [3, 4]]

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

A = [[1, 1], [2, 2], [4, 4]]

```
frequencia = {"Python":1000, "C":750, "Pascal":100, "Java":300}
for linguagem in frequencia :
   if frequencia[linguagem] < 200 :
      print(linguagem, ":", frequencia[linguagem])
frequencia["Python"] = frequencia["Python"] + 100
frequencia["Ada"] = frequencia["Ada"] + 1
print("C :", frequencia["C"])
```

Pascal : 100
KeyError: 'Ada'

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

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

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

10

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

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

TypeError: soma() missing 1 required positional argument: 'c'

```
def subtrai(a, b):
  c = a - b

subtrai(10, 20)
print(c)
```

NameError: name 'c' is not defined

c) (0.9 ponto) **Recursão**

```
def recursiva(n) :
  if n < 1 :
    return n
  else:
    recursiva(n-1)
    print(n * "*")
    return n

recursiva(4)
```

*
**
***
****

```
def recursiva(n) :
  if n < 1 :
    return n
  else:
    recursiva(n-2)
    print(n * "*")
    recursiva(n-2)
    return n

recursiva(5)
```

*
***
*
*****
*
***
*

```
def recursiva(n) :
  if n < 1 :
    return n
  else:
    recursiva(n-2)
    print(n * "*")
    recursiva(n)
    return n

recursiva(5)
```

*
*
...
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.

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \end{bmatrix} \qquad \begin{bmatrix} 0 & 0 & 1 & 1 & 2 & 2 & 3 & 3 \\ 0 & 0 & 1 & 1 & 2 & 2 & 3 & 3 \\ 1 & 1 & 2 & 2 & 3 & 3 & 4 & 4 \\ 1 & 1 & 2 & 2 & 3 & 3 & 4 & 4 \\ 2 & 2 & 3 & 3 & 4 & 4 & 5 & 5 \\ 2 & 2 & 3 & 3 & 4 & 4 & 5 & 5 \\ 3 & 3 & 4 & 4 & 5 & 5 & 6 & 6 \\ 3 & 3 & 4 & 4 & 5 & 5 & 6 & 6 \end{bmatrix}$$

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, 0, 1, 1, 2, 2],
     [0, 0, 1, 1, 2, 2],
     [1, 1, 2, 2, 3, 3],
     [1, 1, 2, 2, 3, 3],
     [2, 2, 3, 3, 5, 5],
     [2, 2, 3, 3, 5, 5]]
```

m[4][4] m[4][5] m[5][4] m[5][5]

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

```
def verifica_padrao(m) :
    for i in range(len(m)):
        for j in range(len(m)):
            if m[i][j] != i // 2 + j // 2:
                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([13, 7, 5, 20, 4, 8, 1, 9])
```

Abaixo, está indicado o que será escrito pela chamada inicial ao comando `print()`. Seguindo o modelo, complete os espaços com o que será escrito pelas próximas cinco chamadas[1].

lista = [	13	7	5	20	4	8	1	9]
lista = [	7	13	5	20	4	8	1	9]
lista = [	5	7	13	20	4	8	1	9]
lista = [	5	7	13	20	4	8	1	9]
lista = [	4	5	7	13	20	8	1	9]
lista = [	4	5	7	8	13	20	1	9]

Qual das frases a seguir melhor reflete o resultado obtido com os primeiros passos do programa?

( ) O maior elemento está sendo deslocado para a última posição da lista.

( ) Os seis primeiros elementos formam uma sublista ordenada com os menores valores da lista.

(**X**) Os seis primeiros elementos formam uma sublista ordenada, mas o menor valor da lista não está neste conjunto.

Qual é o nome do algoritmo implementado? Insertion Sort

[1] **Dica:** O algoritmo precisa de mais de cinco passos para ordenar a lista e, portanto, a lista ainda não estará ordenada na última linha a ser preenchida.

4. (3.0 pontos) Uma partida de um jogo de futebol pode ser descrita com tuplas da seguinte forma:

`(time_casa, gols_casa, time_visitante, gols_visitante)`

Uma lista de tuplas no formato descrito acima pode ser utilizada para armazenar os dados de um campeonato:

`campeonato = [(time`$_0$`, gols`$_0$`, time`$_1$`, gols`$_1$`), (time`$_2$`, gols`$_2$`, time`$_3$`, gols`$_3$`), ...]`

Escreva uma função que recebe o nome de um time e uma lista com tuplas descrevendo as partidas de um campeonato e retorna a **pontuação** deste time no campeonato descrito. Para o cálculo da pontuação, considere que os times recebem três pontos por vitória, um ponto por empate e zero por derrota.

`pontuacao(nome_time, campeonato)`

```
def pontuacao(nome_time, campeonato) :
  n_pontos = 0
  for partida in campeonato :
     if (nome_time == partida[0] and partida[1] > partida[3] or
         nome_time == partida[2] and partida[3] > partida[1]) :
         n_pontos += 3
     elif (partida[1] == partida[3] and
         (nome_time == partida[0] or nome_time == partida[2])) :
         n_pontos += 1

  return n_pontos
```

Exemplo de chamada em que `pontuacao_ibis` deverá receber o valor 4:

```
campeonato = [("Ibis", 0, "Cascavel", 0), ("Porto", 2, "Dourados", 5),
              ("Ibis", 3, "Dourados", 0), ("Cascavel", 1, "Porto", 0)]
pontuacao_ibis = pontuacao("Ibis", campeonato)
```

MC102

Algoritmos e Programação de Computadores

Prova 2

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

TypeError: can only concatenate list (not "int") to list

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

[3, 6, 9, '12.0']

```
lista = [3, 6]
lista[2] = 9
print(lista)
```

IndexError: list assignment index out of range

```
lista = [("A", 1), ("B", 2)]
lista[0] = ["D", 3]
lista[0][1] = 4
print(lista)
```

[['D', 4], ('B', 2)]

```
A = [[1, 2], [3, 4], [5, 6]]

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

A = [[2, 2], [4, 4], [6, 6]]

```
frequencia = {"Python":1000, "C":750, "Ada":100, "Java":300}
for linguagem in frequencia :
   if frequencia[linguagem] < 200 :
      print(linguagem, ":", frequencia[linguagem])
frequencia["Python"] = frequencia["Python"] + 100
frequencia["Pascal"] = frequencia["Pascal"] + 1
print("C :", frequencia["C"])
```

Ada : 100
KeyError: 'Pascal'

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

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

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

TypeError: soma() missing 1 required positional argument: 'c'

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

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

10

```
def subtrai(a, b):
  c = a - b

subtrai(5, 10)
print(c)
```

NameError: name 'c' is not defined

c) (0.9 ponto) **Recursão**

```
def recursiva(n) :
  if n == 0 :
    return n
  else:
    print(n * "*")
    recursiva(n-1)
    return n

recursiva(4)
```

****
***
**
*

```
def recursiva(n) :
  if n == 0 :
    return n
  else:
    recursiva(n-1)
    print(n * "*")
    recursiva(n-1)
    return n

recursiva(3)
```

*
**
*
***
*
**
*

```
def recursiva(n) :
  if n == 0 :
    return n
  else:
    recursiva(n-1)
    print(n * "*")
    recursiva(n)
    return n

recursiva(5)
```

*
*
...
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.

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 1 & 1 & 2 & 2 & 2 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 3 \\ 1 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 3 \\ 1 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 3 \\ 2 & 2 & 2 & 3 & 3 & 3 & 4 & 4 & 4 \\ 2 & 2 & 2 & 3 & 3 & 3 & 4 & 4 & 4 \\ 2 & 2 & 2 & 3 & 3 & 3 & 4 & 4 & 4 \end{bmatrix}$$

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, 0, 0, 1, 1, 1],
     [0, 0, 0, 1, 1, 1],
     [0, 0, 0, 1, 1, 1],
     [3, 3, 3, 2, 2, 2],
     [3, 3, 3, 2, 2, 2],
     [3, 3, 3, 2, 2, 2]]
```

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

Considerando este modelo de representação de matrizes, escreva uma função `verifica_padrao(m)` que retorna `True` se uma matriz quadrada `m` passada como parâmetro respeitar o padrão ou `False` caso contrário. Considere que `m` será uma matriz quadrada e que o valor da altura será múltiplo de 3 . Utilize o comando `len()` para obter as dimensões de `m`.

```
def verifica_padrao(m) :
    for i in range(len(m)):
        for j in range(len(m)):
            if m[i][j] != i // 3 + j // 3:
                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(len(lista)):
     aux = i
     for j in range(i+1, len(lista)):
         if lista[j] < lista[aux]:
             aux = j
     # troca elementos lista[i] e lista[aux]
     lista[i], lista[aux] = lista[aux], lista[i]
     print("lista =", lista)
```

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

```
ordena([13, 7, 5, 20, 4, 8, 1, 9])
```

Abaixo, está indicado o que será escrito pela chamada inicial ao comando `print()`. Seguindo o modelo, complete os espaços com o que será escrito pelas próximas cinco chamadas[1].

lista = [	13	7	5	20	4	8	1	9]
lista = [	1	7	5	20	4	8	13	9]
lista = [	1	4	5	20	7	8	13	9]
lista = [	1	4	5	20	7	8	13	9]
lista = [	1	4	5	7	20	8	13	9]
lista = [	1	4	5	7	8	20	13	9]

Qual das frases a seguir melhor reflete o resultado obtido com os primeiros passos do programa?

( ) O maior elemento está sendo deslocado para a última posição da lista.

(**X**) Os seis primeiros elementos formam uma sublista ordenada com os menores valores da lista.

( ) Os seis primeiros elementos formam uma sublista ordenada, mas o menor valor da lista não está neste conjunto.

Qual é o nome do algoritmo implementado? Selection Sort

[1] **Dica:** O algoritmo precisa de mais de cinco passos para ordenar a lista e, portanto, a lista ainda não estará ordenada na última linha a ser preenchida.

4. (3.0 pontos) Uma partida de um jogo de futebol pode ser descrita com tuplas da seguinte forma:

`(time_casa, gols_casa, time_visitante, gols_visitante)`

Uma lista de tuplas no formato descrito acima pode ser utilizada para armazenar os dados de um campeonato:

`campeonato = [(time`$_0$`, gols`$_0$`, time`$_1$`, gols`$_1$`), (time`$_2$`, gols`$_2$`, time`$_3$`, gols`$_3$`), ...]`

Escreva uma função que recebe o nome de um time e uma lista com tuplas descrevendo as partidas de um campeonato e retorna a **pontuação** deste time no campeonato descrito. Para o cálculo da pontuação, considere que os times recebem três pontos por vitória, um ponto por empate e zero por derrota.

`pontuacao(nome_time, campeonato)`

```
def pontuacao(nome_time, campeonato) :
  n_pontos = 0
  for partida in campeonato :
     if (nome_time == partida[0] and partida[1] > partida[3] or
         nome_time == partida[2] and partida[3] > partida[1]) :
         n_pontos += 3
     elif (partida[1] == partida[3] and
         (nome_time == partida[0] or nome_time == partida[2])) :
         n_pontos += 1

  return n_pontos
```

Exemplo de chamada em que `pontuacao_ibis` deverá receber o valor 4:

```
campeonato = [("Ibis", 0, "Cascavel", 0), ("Porto", 2, "Dourados", 5),
              ("Ibis", 3, "Dourados", 0), ("Cascavel", 1, "Porto", 0)]
pontuacao_ibis = pontuacao("Ibis", campeonato)
```

MC102

Algoritmos e Programação de Computadores

Prova 2

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 = [3, 6]
lista[2] = 9
print(lista)
```

IndexError: list assignment index out of range

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

TypeError: can only concatenate list (not "int") to list

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

[3, 6, 9, '12.0']

```
lista = [("A", 1), ("B", 2)]
lista[1] = ["C", 2]
lista[1][1] = 3
print(lista)
```

[('A', 1), ['C', 3]]

```
A = [[0, 2], [2, 4], [4, 8]]

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

A = [[2, 2], [4, 4], [8, 8]]

```
frequencia = {"Python":1000, "C":750, "Fortran":100, "Java":300}
for linguagem in frequencia :
   if frequencia[linguagem] < 200 :
      print(linguagem, ":", frequencia[linguagem])
frequencia["Python"] = frequencia["Python"] + 100
frequencia["Pascal"] = frequencia["Pascal"] + 1
print("Java :", frequencia["Java"])
```

Fortran : 100
KeyError: 'Pascal'

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

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

c = 5
print(soma(0, 10))
```

TypeError: soma() missing 1 required positional argument: 'c'

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

c = 5
print(soma(0, 10))
```

15

```
def subtrai(a, b):
  c = a - b

subtrai(15, 10)
print(c)
```

NameError: name 'c' is not defined

c) (0.9 ponto) **Recursão**

```
def recursiva(n) :
  if n == 0 :
    return n
  else:
    print(n * "+")
    recursiva(n-1)
    return n

recursiva(4)
```

++++
+++
++
+

```
def recursiva(n) :
  if n == 0 :
    return n
  else:
    recursiva(n-1)
    print(n * "+")
    recursiva(n)
    return n

recursiva(5)
```

+
+
...
RecursionError: maximum recursion depth exceeded in comparison

```
def recursiva(n) :
  if n < 1 :
    return n
  else:
    recursiva(n-2)
    print(n * "+")
    recursiva(n-2)
    return n

recursiva(5)
```

+
+++
+
+++++
+
+++
+

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.

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \\ 2 & 2 & 3 & 3 \\ 2 & 2 & 3 & 3 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 2 & 2 & 3 & 3 & 4 & 4 \\ 1 & 1 & 2 & 2 & 3 & 3 & 4 & 4 \\ 2 & 2 & 3 & 3 & 4 & 4 & 5 & 5 \\ 2 & 2 & 3 & 3 & 4 & 4 & 5 & 5 \\ 3 & 3 & 4 & 4 & 5 & 5 & 6 & 6 \\ 3 & 3 & 4 & 4 & 5 & 5 & 6 & 6 \\ 4 & 4 & 5 & 5 & 6 & 6 & 7 & 7 \\ 4 & 4 & 5 & 5 & 6 & 6 & 7 & 7 \end{bmatrix}$$

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 = [[0, 1, 2, 2, 5, 5],
     [1, 1, 2, 2, 5, 5],
     [2, 2, 3, 3, 4, 4],
     [2, 2, 3, 3, 4, 4],
     [3, 3, 4, 4, 5, 5],
     [3, 3, 4, 4, 5, 5]]
```

m[0][0] m[0][5] m[0][4] m[1][4] m[1][5]

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

```
def verifica_padrao(m) :
    for i in range(len(m)):
        for j in range(len(m)):
            if m[i][j] != i // 2 + j // 2 + 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(len(lista)):
    for j in range(len(lista) - i - 1):
       if lista[j] > lista[j + 1]:
          # troca elementos lista[j] e lista[j + 1]
          lista[j], lista[j + 1] = lista[j + 1], lista[j]
          print("lista =", lista)
```

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

```
ordena([13, 7, 5, 20, 4, 8, 1, 9])
```

Abaixo, está indicado o que será escrito pela chamada inicial ao comando `print()`. Seguindo o modelo, complete os espaços com o que será escrito pelas próximas cinco chamadas[1].

lista = [13 , 7 , 5 , 20 , 4 , 8 , 1 , 9]

lista = [7 , 13 , 5 , 20 , 4 , 8 , 1 , 9]

lista = [7 , 5 , 13 , 20 , 4 , 8 , 1 , 9]

lista = [7 , 5 , 13 , 4 , 20 , 8 , 1 , 9]

lista = [7 , 5 , 13 , 4 , 8 , 20 , 1 , 9]

lista = [7 , 5 , 13 , 4 , 8 , 1 , 20 , 9]

Qual das frases a seguir melhor reflete o resultado obtido com os primeiros passos do programa?

(**X**) O maior elemento está sendo deslocado para a última posição da lista.

( ) Os seis primeiros elementos formam uma sublista ordenada com os menores valores da lista.

( ) Os seis primeiros elementos formam uma sublista ordenada, mas o menor valor da lista não está neste conjunto.

Qual é o nome do algoritmo implementado? Bubble Sort

[1] **Dica:** O algoritmo precisa de mais de cinco passos para ordenar a lista e, portanto, a lista ainda não estará ordenada na última linha a ser preenchida.

4. (3.0 pontos) Uma partida de um jogo de futebol pode ser descrita com tuplas da seguinte forma:

`(time_casa, gols_casa, time_visitante, gols_visitante)`

Uma lista de tuplas no formato descrito acima pode ser utilizada para armazenar os dados de um campeonato:

`campeonato = [(time`$_0$`, gols`$_0$`, time`$_1$`, gols`$_1$`), (time`$_2$`, gols`$_2$`, time`$_3$`, gols`$_3$`), ...]`

Escreva uma função que recebe o nome de um time e uma lista com tuplas descrevendo as partidas de um campeonato e retorna a **pontuação** deste time no campeonato descrito. Para o cálculo da pontuação, considere que os times recebem três pontos por vitória, um ponto por empate e zero por derrota.

`pontuacao(nome_time, campeonato)`

```
def pontuacao(nome_time, campeonato) :
  n_pontos = 0
  for partida in campeonato :
     if (nome_time == partida[0] and partida[1] > partida[3] or
         nome_time == partida[2] and partida[3] > partida[1]) :
         n_pontos += 3
     elif (partida[1] == partida[3] and
         (nome_time == partida[0] or nome_time == partida[2])) :
         n_pontos += 1

  return n_pontos
```

Exemplo de chamada em que `pontuacao_ibis` deverá receber o valor 4:

```
campeonato = [("Ibis", 0, "Cascavel", 0), ("Porto", 2, "Dourados", 5),
              ("Ibis", 3, "Dourados", 0), ("Cascavel", 1, "Porto", 0)]
pontuacao_ibis = pontuacao("Ibis", campeonato)
```