MC102

Algoritmos e Programação de Computadores

Prova 2

Turmas A B C E F G H I

Primeiro Semestre de 2019

Questão	Nota
1	
2	
3	
4	
5	
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. (2.5 pontos) 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, deixe seu espaço de resposta em branco. Caso algum erro seja encontrado, indique o motivo e marque no código o ponto em que ele ocorre.

a) **Funções, passagem de parâmetros e escopo de variáveis**

```
# True or False?
def maior(a, b):
  print(a > b)

maior(10, 20)
```

False

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

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

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

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

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

name 'c' is not defined

```
def soma(a, b) :
  r = a + b
  a = 0
  b = 0
  return r

a = 10
b = 20
c = soma(a, b)
print(a, b, c)
```

10 20 30

```
def misterio(a, b, c):
    if a < b and a < c :
        print(a)
    elif b < c:
        print(b)
    else:
        print(c)
    return

misterio(5, 6, 1)
```

1

```
def menos(a) :
  return -a

def subtrai(a, b) :
    return a + menos(b)

a = 10
b = 20
r = subtrai(a,b)
print(menos(r))
```

10

b) **Listas, tuplas e dicionários**

```
lista = [0, 1, 4, 5, 12]
lista.append(3)
print(lista)
```

[0, 1, 4, 5, 12, 3]

```
lista = [0, 1, 4, 5, 12]
lista[7] = 3
print(lista)
```

list assignment index out of range

```
tupla = ("Ibis", 0, "Porto", 3)
tupla[1] = 10
print(tupla)
```

'tuple' object does not support item assignment

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

[('C', 3), ('B', 2)]

```
A = [[6, 3, 2], [7, 2, 0], [2, 1, 0]]
B = [[1, 2, 4, 5], [2, -2, 0, 1], [1, 0, -1, 4]]

for i in range(len(A)) :
    for j in range(len(A)) :
        A[i][j] = B[i][j] - 1
print(A[0])
```

[0, 1, 3]

```
vitorias = {"Ibis":0, "Porto":1, "Itapipoca":3, "Bonito":1}
for time in vitorias :
    if vitorias[time] > 2 :
        print(time, vitorias[time])
```

Itapipoca 3

```
vitorias = {"Ibis":0, "Porto":1, "Verona":2, "Bonito":1}
partida = ("Itapipoca", 1, "Cruzeiro", 0)
if partida[1] > partida[3] :
    vencedor = partida[0]
else:
    vencedor = partida[2]
vitorias[vencedor] = vitorias[vencedor] + 1
print(vitorias[vencedor])
```

KeyError: 'Itapipoca'

Dica: método `append()` adiciona um elemento ao final de uma lista

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

```
1 1 1
2 2 2
3 3 3
```

```
1 1 1 1
2 2 2 2
3 3 3 3
4 4 4 4
```

```
1 1 1 1 1
2 2 2 2 2
3 3 3 3 3
4 4 4 4 4
5 5 5 5 5
```

Indique a razão pela qual a matriz a seguir não segue este padrão:

```
1 1 1 1 1 1
2 2 2 2 2 2
3 3 3 3 3 3
4 4 4 4 4 4
5 5 5 5 5 5
6 0 6 0 6 6
```

Todos os valores da última linha deveriam ser iguais a 6.

Escreva uma função `verifica_padrao(m)` que retorna `True` se uma matriz quadrada `m` passada como parâmetro respeita o padrão ou `False` caso contrário. Não é necessário testar se a matriz `m` é quadrada.

```
def verifica_padrao(m) :
    for i in range(len(m)):
        for j in range(len(m)):
            if m[i][j] != i + 1:
                return False
    return True
```

3. (2.5 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([12, 7, 9, 1, 3, 8, 20, 6])
```

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 = [12 , 7 , 9 , 1 , 3 , 8 , 20 , 6]

lista = [7 , 12 , 9 , 1 , 3 , 8 , 20 , 6]

lista = [7 , 9 , 12 , 1 , 3 , 8 , 20 , 6]

lista = [1 , 7 , 9 , 12 , 3 , 8 , 20 , 6]

lista = [1 , 3 , 7 , 9 , 12 , 8 , 20 , 6]

lista = [1 , 3 , 7 , 8 , 9 , 12 , 20 , 6]

Qual das frases a seguir melhor define o comportamento do laço interno do programa?

( ) O maior elemento é deslocado para a última posição de uma sublista, via uma sequência de trocas.

(**X**) Um novo elemento é inserido em uma sublista ordenada.

( ) O primeiro e o menor elemento de uma sublista trocam de posição.

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. (2.0 pontos) João estava estudando recursão. Dada uma função recursiva simples, ele gostaria de desenvolver uma versão iterativa que produzisse exatamente a mesma saída e o mesmo valor de retorno. Infelizmente, para a função `recursiva()`, a sua `primeira_tentativa()` não produziu os resultados esperados. Mostre isso, indicando o que será escrito pelos programas abaixo:

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

r = recursiva(3)
print("Resultado =", r)
```

```
3
2
1
0
Resultado = 6
```

```
def primeira_tentativa(n) :
    aux = 0
    for i in range(n) :
        print(aux)
        aux = aux + i
    return aux

r = primeira_tentativa(3)
print("Resultado =", r)
```

```
0
0
1
Resultado = 3
```

Implemente a função `iterativa(n)` de maneira a produzir a mesma saída e o mesmo valor de retorno que a função `recursiva(n)`. Considere que as funções receberão sempre valores inteiros maiores ou iguais a zero como parâmetro.

```
def iterativa(n) :
    aux = 0
    for i in range(n, -1, -1) :
        print(i)
        aux += i
    return aux
```

5. (3.0 pontos) As tarefas de laboratório de MC202 têm pesos diferentes. De acordo com o critério de aprovação, um(a) aluno(a) precisa ter **média ponderada mínima** 5.0 e nota mínima 5.0 na **última tarefa** para poder ser aprovado(a) sem exame. Escreva uma função

`media_labs_suficiente(labs)`

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

`labs = [ (nota`$_0$`, peso`$_0$`), (nota`$_1$`, peso`$_1$`), ..., (nota`$_{n-1}$`, peso`$_{n-1}$`) ]`

```
def media_labs_suficiente(labs) :
  peso_labs = 0
  nota_labs = 0
  for lab in labs :
     nota_labs += lab[0] * lab[1]
     peso_labs += lab[1]

  media = nota_labs / peso_labs

  return media >= 5.0 and nota_labs[-1][0] >= 5.0
```

MC102

Algoritmos e Programação de Computadores

Prova 2

Turmas A B C E F G H I

Primeiro Semestre de 2019

Questão	Nota
1	
2	
3	
4	
5	
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. (2.5 pontos) 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, deixe seu espaço de resposta em branco. Caso algum erro seja encontrado, indique o motivo e marque no código o ponto em que ele ocorre.

a) **Funções, passagem de parâmetros e escopo de variáveis**

```
# True or False?
def maior(a, b):
  print(a > b)

maior(20, 10)
```

True

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

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

name 'c' is not defined

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

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

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

```
def soma(a, b) :
  r = a + b
  a = 0
  b = 0
  return r

a = 5
b = 10
c = soma(a, b)
print(a, b, c)
```

5 10 15

```
def misterio(a, b, c):
    if a < b and a < c :
        print(a)
    elif b < c:
        print(b)
    else:
        print(c)
    return

misterio(5, 6, 3)
```

3

```
def menos(a) :
  return -a

def subtrai(a, b) :
    return a + menos(b)

a = 5
b = 10
r = subtrai(a,b)
print(menos(r))
```

5

b) **Listas, tuplas e dicionários**

```
lista = [0, 2, 5, 7, 12]
lista.append(15)
print(lista)
```

[0, 2, 5, 7, 12, 15]

```
lista = [0, 2, 5, 7, 12]
lista[7] = 3
print(lista)
```

list assignment index out of range

```
tupla = ("Ibis", 0, "Porto", 3)
tupla[1] = 10
print(tupla)
```

'tuple' object does not support item assignment

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

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

```
A = [[6, 3, 2], [7, 2, 0], [2, 1, 0]]
B = [[2, 3, 5, 7], [2, -2, 0, 1], [1, 0, -1, 4]]

for i in range(len(A)) :
    for j in range(len(A)) :
        A[i][j] = B[i][j] - 1
print(A[0])
```

[1, 2, 4]

```
vitorias = {"Ibis":4, "Porto":1, "Itapipoca":3, "Bonito":1}
for time in vitorias :
    if vitorias[time] > 3 :
        print(time, vitorias[time])
```

Ibis 4

```
vitorias = {"Ibis":0, "Porto":1, "Verona":2, "Bonito":1}
partida = ("Itapipoca", 1, "Cruzeiro", 3)
if partida[1] > partida[3] :
    vencedor = partida[0]
else:
    vencedor = partida[2]
vitorias[vencedor] = vitorias[vencedor] + 1
print(vitorias[vencedor])
```

KeyError: 'Cruzeiro'

Dica: método `append()` adiciona um elemento ao final de uma lista

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

```
0 1 2
0 1 2
0 1 2
```

```
0 1 2 3
0 1 2 3
0 1 2 3
0 1 2 3
```

```
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
0 1 2 3 4
```

Indique a razão pela qual a matriz a seguir não segue este padrão:

```
0 1 2 3 4 2
0 1 2 3 4 5
0 1 2 3 4 5
0 1 2 3 4 5
0 1 2 3 4 5
0 1 2 3 4 5
```

Todos os valores da última coluna deveriam ser iguais a 5.

Escreva uma função `verifica_padrao(m)` que retorna `True` se uma matriz quadrada `m` passada como parâmetro respeita o padrão ou `False` caso contrário. Não é necessário testar se a matriz `m` é quadrada.

```
def verifica_padrao(m) :
    for i in range(len(m)):
        for j in range(len(m)):
            if m[i][j] != j :
                return False
    return True
```

3. (2.5 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([12, 7, 9, 1, 3, 8, 20, 6])
```

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 = [12 | 7 | 9 | 1 | 3 | 8 | 20 | 6]

lista = [1 | 7 | 9 | 12 | 3 | 8 | 20 | 6]

lista = [1 | 3 | 9 | 12 | 7 | 8 | 20 | 6]

lista = [1 | 3 | 6 | 12 | 7 | 8 | 20 | 9]

lista = [1 | 3 | 6 | 7 | 12 | 8 | 20 | 9]

lista = [1 | 3 | 6 | 7 | 8 | 12 | 20 | 9]

Qual das frases a seguir melhor define o comportamento do laço interno do programa?

( ) O maior elemento é deslocado para a última posição de uma sublista, via uma sequência de trocas.

( ) Um novo elemento é inserido em uma sublista ordenada.

(X) O primeiro e o menor elemento de uma sublista trocam de posição.

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. (2.0 pontos) João estava estudando recursão. Dada uma função recursiva simples, ele gostaria de desenvolver uma versão iterativa que produzisse exatamente a mesma saída e o mesmo valor de retorno. Infelizmente, para a função `recursiva()`, a sua `primeira_tentativa()` não produziu os resultados esperados. Mostre isso, indicando o que será escrito pelos programas abaixo:

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

r = recursiva(3)
print("Resultado =", r)
```

```
3
2
1
0
Resultado = 7
```

```
def primeira_tentativa(n) :
    aux = 1
    for i in range(n) :
        print(aux)
        aux = aux + i
    return aux

r = primeira_tentativa(3)
print("Resultado =", r)
```

```
1
1
2
Resultado = 4
```

Implemente a função `iterativa(n)` de maneira a produzir a mesma saída e o mesmo valor de retorno que a função `recursiva(n)`. Considere que as funções receberão sempre valores inteiros maiores ou iguais a zero como parâmetro.

```
def iterativa(n) :
    aux = 1
    for i in range(n, -1, -1) :
        print(i)
        aux += i
    return aux
```

5. (3.0 pontos) As tarefas de laboratório de MC202 têm pesos diferentes. De acordo com o critério de aprovação, um(a) aluno(a) precisa ter **média ponderada mínima** 5.0 e nota mínima 5.0 na **última tarefa** para poder ser aprovado(a) sem exame. Escreva uma função

`media_labs_suficiente(labs)`

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

`labs = [ (nota`$_0$`, peso`$_0$`), (nota`$_1$`, peso`$_1$`), ..., (nota`$_{n-1}$`, peso`$_{n-1}$`) ]`

```
def media_labs_suficiente(labs) :
  peso_labs = 0
  nota_labs = 0
  for lab in labs :
     nota_labs += lab[0] * lab[1]
     peso_labs += lab[1]

  media = nota_labs / peso_labs

  return media >= 5.0 and nota_labs[-1][0] >= 5.0
```

MC102

Algoritmos e Programação de Computadores

Prova 2

Turmas A B C E F G H I

Primeiro Semestre de 2019

Questão	Nota
1	
2	
3	
4	
5	
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. (2.5 pontos) 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, deixe seu espaço de resposta em branco. Caso algum erro seja encontrado, indique o motivo e marque no código o ponto em que ele ocorre.

a) **Funções, passagem de parâmetros e escopo de variáveis**

```
# True or False?
def menor(a, b):
  print(a < b)

menor(10, 20)
```

True

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

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

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

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

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

c

```
def soma(a, b) :
  r = a + b
  a = 0
  b = 0
  return r

a = 1
b = 2
c = soma(a, b)
print(a, b, c)
```

1 2 3

```
def misterio(a, b, c):
    if a > b and a > c :
        print(a)
    elif b > c:
        print(b)
    else:
        print(c)
    return

misterio(5, 6, 1)
```

6

```
def menos(a) :
  return -a

def subtrai(a, b) :
    return a + menos(b)

a = 4
b = 8
r = subtrai(a,b)
print(menos(r))
```

4

b) **Listas, tuplas e dicionários**

```
lista = [0, 1, 4, 5, 12]
lista[7] = 3
print(lista)
```

list assignment index out of range

```
lista = [0, 1, 4, 5, 12]
lista.append(3)
print(lista)
```

[0, 1, 4, 5, 12, 3]

```
tupla = ("Ibis", 0, "Porto", 3)
tupla[1] = 10
print(tupla)
```

'tuple' object does not support item assignment

```
lista_tuplas = [("A", 1),("B",2)]
lista_tuplas[0] = ("C", 0)
print(lista_tuplas)
```

[('C', 0), ('B', 2)]

```
A = [[6, 3, 2], [7, 2, 0], [2, 1, 0]]
B = [[4, 5, 6, 7], [2, -2, 0, 1], [1, 0, -1, 4]]

for i in range(len(A)) :
    for j in range(len(A)) :
        A[i][j] = B[i][j] - 1
print(A[0])
```

[3, 4, 5]

```
vitorias = {"Ibis":0, "Porto":1, "Itapipoca":2, "Bonito":4}
for time in vitorias :
    if vitorias[time] > 2 :
        print(time, vitorias[time])
```

Bonito 4

```
vitorias = {"Ibis":0, "Porto":1, "Verona":2, "Bonito":1}
partida = ("Caucaia", 1, "Porto", 0)
if partida[1] > partida[3] :
    vencedor = partida[0]
else:
    vencedor = partida[2]
vitorias[vencedor] = vitorias[vencedor] + 1
print(vitorias[vencedor])
```

KeyError: 'Caucaia'

Dica: método `append()` adiciona um elemento ao final de uma lista

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

```
2 2 2
1 1 1
0 0 0
```

```
3 3 3 3
2 2 2 2
1 1 1 1
0 0 0 0
```

```
4 4 4 4 4
3 3 3 3 3
2 2 2 2 2
1 1 1 1 1
0 0 0 0 0
```

Indique a razão pela qual a matriz a seguir não segue este padrão:

```
5 5 5 5 5 5
4 4 4 4 4 4
3 3 3 3 3 3
2 2 2 2 2 2
1 1 1 1 1 1
2 0 0 0 0 0
```

Todos os valores da última linha deveriam ser iguais a 0.

Escreva uma função `verifica_padrao(m)` que retorna `True` se uma matriz quadrada `m` passada como parâmetro respeita o padrão ou `False` caso contrário. Não é necessário testar se a matriz `m` é quadrada.

```
def verifica_padrao(m) :
    for i in range(len(m)):
        for j in range(len(m)):
            if m[i][j] != len(m) - i - 1:
                return False
    return True
```

3. (2.5 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(0, 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([12, 7, 9, 1, 3, 8, 20, 6])
```

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 = [12 | 7 | 9 | 1 | 3 | 8 | 20 | 6]

lista = [7 | 12 | 9 | 1 | 3 | 8 | 20 | 6]

lista = [7 | 9 | 12 | 1 | 3 | 8 | 20 | 6]

lista = [7 | 9 | 1 | 12 | 3 | 8 | 20 | 6]

lista = [7 | 9 | 1 | 3 | 12 | 8 | 20 | 6]

lista = [7 | 9 | 1 | 3 | 8 | 12 | 20 | 6]

Qual das frases a seguir melhor define o comportamento do laço interno do programa?

(**X**) O maior elemento é deslocado para a última posição de uma sublista, via uma sequência de trocas.

( ) Um novo elemento é inserido em uma sublista ordenada.

( ) O primeiro e o menor elemento de uma sublista trocam de posição.

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. (2.0 pontos) João estava estudando recursão. Dada uma função recursiva simples, ele gostaria de desenvolver uma versão iterativa que produzisse exatamente a mesma saída e o mesmo valor de retorno. Infelizmente, para a função `recursiva()`, a sua `primeira_tentativa()` não produziu os resultados esperados. Mostre isso, indicando o que será escrito pelos programas abaixo:

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

r = recursiva(3)
print("Resultado =", r)
```

```
3
2
1
Resultado = 6
```

```
def primeira_tentativa(n) :
    aux = 1
    for i in range(n) :
        print(aux)
        aux = aux * i
    return aux

r = primeira_tentativa(3)
print("Resultado =", r)
```

```
1
0
0
Resultado = 0
```

Implemente a função `iterativa(n)` de maneira a produzir a mesma saída e o mesmo valor de retorno que a função `recursiva(n)`. Considere que as funções receberão sempre valores inteiros maiores ou iguais a zero como parâmetro.

```
def iterativa(n) :
    aux = 1
    for i in range(n, 0, -1) :
        print(i)
        aux *= i
    return aux
```

5. (3.0 pontos) As tarefas de laboratório de MC202 têm pesos diferentes. De acordo com o critério de aprovação, um(a) aluno(a) precisa ter **média ponderada mínima** 5.0 e nota mínima 5.0 na **última tarefa** para poder ser aprovado(a) sem exame. Escreva uma função

`media_labs_suficiente(labs)`

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

`labs = [ (nota`$_0$`, peso`$_0$`), (nota`$_1$`, peso`$_1$`), ..., (nota`$_{n-1}$`, peso`$_{n-1}$`) ]`

```
def media_labs_suficiente(labs) :
  peso_labs = 0
  nota_labs = 0
  for lab in labs :
     nota_labs += lab[0] * lab[1]
     peso_labs += lab[1]

  media = nota_labs / peso_labs

  return media >= 5.0 and nota_labs[-1][0] >= 5.0
```