MAC2166 - Introducao a Computacao - 2017S1 **Avaliacao 2**

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 para marcar as caixas e preencha a caixa **totalmente** para correta interpretação. Exemplo: ■. **Não use** ☒.

Turma (marque somente um número, consulte seu professor se não souber):

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

Marque as caixas ao lado para formar o seu NUSP (coloque um zero à esquerda se o seu NUSP tem só 7 dígitos) e escreva na caixa abaixo seu NUSP e nome em letra legível.

NUSP: | Nome (completo):

.....

Esta avaliação tem duração de 120 minutos. Não desmonte este caderno.

Q1 [1 ponto] Simule o código abaixo e selecione a opção correspondente à saída impressa do programa.

```
def main() :
    L0 = []
    v = 32
    while (v > 0) :
        v = (v // 2) - 2
        L0.append([v, 2 * v])
        L1 = L0
        L1[0][1] = (L1[0][0] + L1[len(L1) -1][0]) // 2
    print(L0)
main()
```

Rascunho

- ☐ `[[0, 0]]`
- ☐ `[[30, 60], [15, 30], [7, 14], [3, 6]]`
- ☐ `[[30, 15], [10, 5], [6, 3], [2, 1]]`
- ☐ `[[14, 28], [5, 10], [0, 0]]`
- ☐ `[[32, 16], [8, 4], [2, 1], [0, 0]]`
- ☐ `[[16, 16], [8, 8], [4, 4], [2, 2], [1, 1]]`
- ☐ `[[14, 7], [5, 2], [3, 1], [0, 0]]`
- ☐ `[[14, 0], [0, 14]]`
- ☐ `[[14, 7], [10, 5], [ 6, 3], [ 2, 1]]`
- ☐ `[[14, 28], [7, 14], [3, 7], [0, 0]]`
- ☐ `[14, 7, 3, 0]`
- ☐ `[[32, 14], [14, 5], [5, 1], [0, 0]]`
- ☐ `[[14, 7], [ 5, 10], [ 0, 0]]`
- ☐ `[[14, 7], [7, 3], [3, 1], [1, 1]]`

Q2 [3 pontos] Considere os 4 seguintes trechos de um mesmo programa (T1 até T4) e depois selecione as afirmações verdadeiras sobre eles pintando as quadrículas. **Considerações**: **1.** As opções sobre cada trecho podem conter desde nenhuma afirmação correta até todas. **2.** A cada item errado que for selecionado, desconta-se nota do exercício.

T1:
```
def funcA(L) :
    v = 0
    i = 0
    while i < len(L) :
        if v < L[i] :
            v = L[i]
        i += 1
    return v
```

T2:
```
def funcB(L) :
    v = L[0]
    i = 1
    while i < len(L) :
        if v > L[i] :
            v = L[i]
        i += 1
    return v
```

T3:
```
def funcC(v,L) :
    i = 0
    while i < len(L) and v != L[i] :
        i += 1
    if i < len(L) :
        R = i
    else :
        R = -1
    return R
```

T4:
```
def main() :
    N = [0,1,3,2,4,7,5]
    A = funcA(N)
    B = funcB(N)
    iA = funcC(A,N)
    iB = funcC(B,N)
    print(A,B,iA,iB)
```

T1
- ☐ Retorna o maior valor em L se ela contém apenas naturais
- ☐ Dá erro de execução se L tiver repetições de valores
- ☐ Dá erro de execução se L for vazia
- ☐ Não retorna o maior valor em L se ela contém apenas negativos

T2
- ☐ Dá erro de execução se L for vazia
- ☐ Retorna o menor valor em L se ela contém apenas naturais
- ☐ Dá erro de execução se L contém valores *float*
- ☐ Não retorna o menor valor em L se ela contém apenas negativos

T3
- ☐ Para todo v e toda lista L, retorna a posição do maior valor em L
- ☐ Retorna -1 apenas quando L é vazia
- ☐ Dá erro de execução se L for vazia
- ☐ Retorna a posição da primeira ocorrência de v em L

T4
- ☐ Mostra a posição do maior natural em N
- ☐ Mostra o maior natural em N
- ☐ Determina se N é crescente
- ☐ Não mostra a posição do menor natural em N

Q3 [3 pontos] Faça um programa em Python, preenchendo as lacunas (L1 até L12) no código abaixo, que lê uma lista de `n` (sendo `n >= 1`) números reais e a coloca em ordem crescente, mostrando o resultado no final.

```
def encontra_menor(A,p):
    ''' Devolve o índice do menor valor em A[p], A[p+1],
        ..., A[n] (sendo n a última posição da lista A) '''
    L1
    for L2:
        if L3:
            L4
    return imenor

def main():
    n = int(input("Digite n (n > 0): "))
    B = []
    for L5:
        x = float(input("Digite um número: "))
        L6
    L7
    while L8:
        m = L9
        menor = B[m]
        L10
        L11
        L12
    print("Lista ordenada:", L)

main()
```

Rascunho

Considere os **12** itens seguintes, correspondentes respectivamente a cada uma das 12 lacunas no código acima. Cada item tem 5 opções, selecione aquela que torna o programa acima correto.

L1:
A `imenor = 1` B `imenor = 0` C `imenor = p` D `imenor = len(A)-1`

L2:
A `i in range(len(A)-1)` B `i in range(p+1, len(A)-1)` C `i in range(len(A))`
D `i in range(p+1, len(A))`

L3:
A `A[i] > A[p]` B `A[i] > A[imenor]` C `A[i] < A[imenor]` D `A[i] < A[p]`

L4:
A `imenor = i` B `imenor = A[i]` C `imenor = p` D `imenor = A[p]`

L5:
A `k in range(0,n-1)` B `k in range(1,n)` C `k in range(1,n+1)` D `k in range(0,n)`

L6:
A `B.append(x)` B `B[k].append(x)` C `B[k] = x` D `B + [x]`

L7:
A `pos = 0` B `pos = 1` C `pos = n` D `pos = n-1`

L8:
A `pos >= 0` B `pos > 0` C `pos < n` D `pos <= n`

L9:
A `encontra_menor(B,pos)` B `encontra_menor(B,n)` C `encontra_menor(B,pos+1)`
D `encontra_menor(B,0)`

L10:
A `B[m] = pos` B `B[m] = B[pos]` C `B[pos] = menor` D `B[pos] = B[m]`

L11:
A `menor = B[pos]` B `B[pos] = menor` C `B[m] = B[pos]` D `B[pos] = B[m]`

L12:
A `pos = pos - 1` B `pos = pos + 1` C `pos = m - 1` D `pos = m + 1`

Q4 [3 pontos] Nesta questão você deve elaborar um programa que, dada uma matriz (= uma lista de listas) onde cada linha é uma lista com o nome de um aluno e as notas que ele tirou em cada avaliação, mostre a matriz com uma coluna adicional com a nota média de cada aluno e uma linha adicional com a nota média de cada avaliação. Considere que na matriz dada há pelo menos uma linha e pelo menos uma coluna de nota (além da primeira coluna de nome). **Exemplo:** para a matriz de notas M1 abaixo, a matriz mostrada pelo programa na saída deve ser a M2.

M1 =

"Ana"	8.5	2.5	4.9
"Igor"	7.3	7.0	6.7
"Luiz"	1.6	2.4	9.2

M2 =

"Ana"	8.5	2.5	4.9	5.3
"Igor"	7.3	7.0	6.7	7.0
"Luiz"	1.6	2.4	9.2	4.4
"Médias"	5.8	3.9	6.9	5.6

Você deve criar e usar duas funções auxiliares, a `mediaPorAluno` e a `mediaPorAvaliacao`, que recebem uma matriz com notas de alunos (como a M1 do exemplo). A primeira acrescenta na matriz uma coluna com a média das notas de cada aluno, enquanto a segunda acrescenta nela uma nova linha (como a última linha de M2) com a nota média de cada coluna de nota. Para elaborar o programa completo, você deve usar apenas trechos de código listados abaixo. Assinale a alternativa que contém os trechos corretos na ordem correta. ATENÇÃO: Os blocos não estão indentados. Assim, o final do laço é indicado por um '*' após o número de bloco que contém o último comando do laço. O final de uma função é indicado por '**' após o número de bloco que contém o último comando da função.

```
# trecho 1
for i in range(numNotas):
  somaAluno += aluno[i]
```

```
#trecho 2
i = 1
while i < len(aluno):
  somaAluno += aluno[i]
  i += 1
```

```
# trecho 3
for i in range(numNotas-1):
  somaAluno += aluno[i+1]
```

```
# trecho 4
numNotas = len(notasAlunos)-1
somaAluno = 0
for aluno in notasAlunos:
```

```
# trecho 5
numNotas = len(notasAlunos[0])-1
for aluno in notasAlunos:
  somaAluno = 0
```

```
#trecho 6
mediaPorAvaliacao(alunos)
mediaPorAluno(alunos)
print(alunos)
```

```
#trecho 7
mediaPorAluno(alunos)
mediaPorAvaliacao(alunos)
print(alunos)
```

```
# trecho 8
def mediaPorAvaliacao(notasAlunos):
  linhaMedias = ['Medias:']
  numAlunos = len(notasAlunos)
```

```
#trecho 9
def main():
  alunos = [["Ana", 8.5,2.5,4.9],
            ["Igor",7.3,7.0,6.7],
            ["Luiz",1.6,2.4,9.2]]
```

```
# trecho 10
def mediaPorAluno(notasAlunos):
```

```
# trecho 11
notasAlunos.append(linhaMedias)
```

```
# trecho 12
notasAlunos[lin+1] = linhaMedias
```

```
# trecho 13
notasAlunos[lin].append(linhaMedias)
```

```
# trecho 14
mediaColuna = somaColuna/numAlunos
linhaMedias[col] = mediaColuna
```

```
# trecho 15
mediaColuna = somaColuna/numAlunos
linhaMedias.append(mediaColuna)
```

```
#trecho 16
main()
```

```
# trecho 17
somaColuna = 0
col = 0
while col < len(notasAlunos[0]):
  col += 1
```

```
# trecho 18
col = 0
while col < len(notasAlunos[0]):
  col += 1
  somaColuna = 0
```

```
# trecho 19
for col in range(1,len(notasAlunos[0])):
  somaColuna = 0
```

```
#trecho 20
mediaAluno = somaAluno/numNotas
aluno[i] = mediaAluno
```

```
#trecho 21
mediaAluno = somaAluno/numNotas
aluno.append(mediaAluno)
```

```
# trecho 22
lin = 0
while lin < numAlunos:
  somaColuna += notasAlunos[lin][col]
  lin += 1
```

```
# trecho 23
for lin in range(1,numAlunos):
  soma += aluno[col]
```

- ☐ `10,4,1,*,*,20,**,8,17,23,*,15,*,12,**,9,6,**,16`
- ☐ `10,4,3,*,20,*,**,8,18,22,*,13,*,12,**,9,7,**,16`
- ☐ `10,5,2,*,21,*,**,8,19,22,*,15,*,11,**,9,6,**,16`
- ☐ `10,5,1,*,21,*,**,8,17,22,*,13,11,*,**,9,6,**,16`
- ☐ `10,5,2,*,*,21,**,8,19,23,*,15,11,*,**,9,7,**,16`
- ☐ `10,4,3,*,20,*,**,8,18,22,*,15,12,*,**,9,6,**,16`
- ☐ `10,4,2,*,20,*,**,8,17,23,*,14,*,11,**,9,7,**,16`
- ☐ `10,5,1,*,21,*,**,8,18,23,*,13,*,11,**,9,7,**,16`
- ☐ `10,5,2,*,21,*,**,8,19,22,*,15,*,11,**,9,7,**,16`
- ☐ `10,4,3,*,21,*,**,8,17,23,*,14,11,*,**,9,7,**,16`

Rascunho