Algoritmos e Programação de Computadores

Correção da Prova 2

Profa. Sandra Avila
Instituto de Computação (IC/Unicamp)

MC102, 28 Junho, 2019

AVISO: Temos 3 tipos de provas.

Questão 1
(Tipo 1)

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)
```

Erro. A função `soma` exige 3 argumentos.

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

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

Erro. `c` não foi definido.

```
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)
```

Erro. A posição 7 não existe.

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

Erro. Tupla é imutável.

```
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])
```

Não é possível acessar `vitorias["Itapipoca"]`.

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

Questão 2
(Tipo 1)

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

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

Indique a razão pela qual a matriz a seguir, implementada em Python por uma lista de listas, 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.

Considerando este modelo de representação de matrizes, escreva uma função `verifica_padrao(m)` que retorna `True` se uma matriz quadrada `m` passada como parâmetro respeita o padrão ou `False` caso contrário. Utilize o comando `len()` para verificar as dimensões da matriz; não é necessário testar se `m` é uma a matriz 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
```

Questão 3
(Tipo 1)

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

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

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

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 , 9 , 1 , 4 , 8 , 20 , 5]

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

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

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

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

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

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.

Questão 4

(Tipo 1, 2 e 3)

4. (2.0 pontos) Amanda estava estudando recursão. Ela descobriu que no século 18 um menino chamado Gauss, aos 8 anos, identificou que a soma dos `N` primeiros números naturais é dada pela fórmula:

$$S(N) = \frac{N*(N+1)}{2}$$

Supondo que Gauss soubesse programar em Python, indique como ele implementaria funções que recebem como parâmetro um número `N` e retornam soma dos `N` primeiros números naturais:

- Sem usar recursão e utilizando a fórmula $S(N) = \frac{N*(N+1)}{2}$.

```
def somaN(N):
    return N * (N+1) // 2
```

- Utilizando recursão, sem utilizar a fórmula $S(N) = \frac{N*(N+1)}{2}$.

```
def somaN(N):
    if N == 0:
        return 0
    return N + somaN(N-1)
```

- Sem utilizar recursão e sem utilizar a fórmula $S(N) = \frac{N*(N+1)}{2}$.

  Utilize um comando repetitivo e uma variável acumuladora.

```
def somaN(N):
    soma = 0
    for i in range(N+1):
        soma = soma + i
    return soma
```

Questão 5

(Tipo 1, 2 e 3)

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** 3.0 em **todas** tarefas para poder ser aprovado(a) sem exame. Escreva uma função

`desempenho_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 desempenho_labs_suficiente(labs):
    peso_labs = 0
    nota_labs = 0
    for lab in labs:
        if lab[0] < 3.0:
            return False
        nota_labs += lab[0] * lab[1]
        peso_labs += lab[1]
    media = nota_labs / peso_labs
    return media >= 5.0
```

Eu odeio programar

Eu odeio programar

Funcionou Aaaaaaaaahhhhhhh!!!!

Eu amo programar

Eu amo programar

...

Obrigada, Pessoal!