MC102 — Algoritmos e Programação de Computadores

GABARITO Teste Conceitual **VERSÃO 3** — Segundo Semestre de 2019

Turmas ABCD EF IJKL MOP QT UVX Z

A entrega deste caderno de questões e da folha avulsa de respostas é obrigatória!!!

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 marcadas na folha avulsa a caneta azul ou preta. Todas as questão terão o mesmo peso. Quatro respostas erradas irão anular uma correta. Questões em branco não são consideradas incorretas, mas não valem pontos.

Bom teste!

1. Qual dos códigos abaixo escreve apenas o valor total da soma de todos os inteiros entre 0 (inclusive) a 9 (inclusive)?

(a)

```
soma = 0
for i in range(1,11):
  soma = soma + i
print(soma)
```

Saída:
55

(b)

```
soma = 0
for i in range(9):
  soma = soma + i
print(soma)
```

Saída:
36

(c)

```
i = 0
soma = 0
while i < 10:
    soma = soma + i
    i = 1
print(soma)
```

Saída:
Loop infinito
...

(d) **Correta**

```
i = 0
soma = 0
while i < 10 :
  soma = soma + i
  i = i + 1
print(soma)
```

Saída:
45

(e)

```
soma = 0
for i in range(10):
  soma = soma + i
  print(soma)
```

Saída:
0
1
3
6
10
15
21
28
36
45

2. Qual dos códigos abaixo irá escrever a lista `[24, 30, 35]`?

(a)

```
a = [4, 10, 15]
x = 20
l = []
for i in a:
  l.append(a[i] + x)
print(l)
```

Saída:
```
IndexError: list
index out of range
```

(b)

```
a = [4, 10, 15]
x = 20
soma = []
for i in range(len(a)):
  soma = soma + a
print(soma)
```

Saída:
```
[4, 10, 15, 4, 10,
15, 4, 10, 15]
```

(c) **Correta**

```
a = [4, 10, 15]
x = 20
l = []
for i in
range(len(a)):
  l.append(a[i]+x)
print(l)
```

Saída:
```
[24, 30, 35]
```

(d)

```
a = [4, 10, 15]
x = 20
l = []
for i in range(len(a)):
  l.append(i+x)
print(l)
```

Saída:
```
[20, 21, 22]
```

(e)

```
a = [4, 10, 15]
x = 20
l = []
for i in a:
  l = l + i + x
print(l)
```

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

Devido à presença de uma quebra de linha adicional na alternativa com código correto, respostas em branco foram pontuadas como corretas.

3. Qual dos códigos abaixo escreve apenas os números inteiros entre `menor` (exclusive) e `maior` (exclusive)? Considere que `menor` < `maior`.

(a) **Correta**

```
menor = int(input())
maior = int(input())
i = menor + 1
while i < maior:
  print(i)
  i = i + 1
```

Entrada:
```
5
10
```
Saída:
```
6
7
8
9
```

(b)

```
menor = int(input())
maior = int(input())
i = menor + 1
if i < maior:
   print(i)
   i = i + 1
```

Entrada:
```
5
10
```
Saída:
```
6
```

(c)

```
menor = int(input())
maior = int(input())
i = menor
while i < maior:
  print(i)
  i = i + 1
```

Entrada:
```
5
10
```
Saída:
```
5
6
7
8
9
```

(d)

```
menor = int(input())
maior = int(input())
i = menor + 1
while i < maior:
  print(i)
  i = menor + 1
```

Entrada:
```
5
10
```
Saída:
```
Loop infinito
6
...
```

(e)

```
menor = int(input())
maior = int(input())
i = menor
maior = menor + 1
while i < maior:
  print(i)
  i = i + 1
```

Entrada:
```
5
10
```
Saída:
```
5
```

4. Um número positivo `n` maior do que 1 é primo se ele possui exatamente 2 divisores: 1 e `n`. Uma forma simples, embora ineficiente, de determinar se `n` é primo é verificar que ele não tem divisores entre 2 (inclusive) e `n-1` (inclusive). Qual dos códigos abaixo implementa corretamente esta abordagem? A função `eh_primo()` deve retornar `True` se o número `n` passado como parâmetro for primo ou `False` caso contrário. Consider `n > 1`.

(a)

```
def eh_primo(n):
  for i in range(2,n):
    if n % i == 0:
      return False
    else:
      return True
```

Teste:
```
print(eh_primo(7))
print(eh_primo(9))
```
Saída:
```
True
True
```

(b)

```
def eh_primo(n):
  i = 2
  if n % i == 0:
    return False
  else:
    return True
```

Teste:
```
print(eh_primo(7))
print(eh_primo(9))
```
Saída:
```
True
True
```

(c)

```
def eh_primo(n):
  for i in range(2,n+1):
    if n % i == 0:
      return False
  return True
```

Teste:
```
print(eh_primo(7))
print(eh_primo(8))
```
Saída:
```
False
False
```

(d)

```
def eh_primo(n):
  for i in range(1,n):
    if n % i == 0:
      return False
  return True
```

Teste:
```
print(eh_primo(7))
print(eh_primo(8))
```
Saída:
```
False
False
```

(e) **Correta**

```
def eh_primo(n):
  for i in range(2,n):
    if n % i == 0:
      return False
  return True
```

Teste:
```
print(eh_primo(7))
print(eh_primo(8))
```
Saída:
```
True
False
```

5. Qual dos códigos abaixo escreve os números pares entre 0 (inclusive) e `n` (exclusive)? Considere zero como par.

(a)

```
n = int(input())
l = list(range(0,2*n,2))
for i in l:
  if l[i] < n:
    print(i)
```

Entrada:
```
6
```
Saída:
```
0
2
IndexError:  list
index out of range
```

(b) **Correta**

```
n = int(input())
i = 0
while i < n:
  print (i)
  i = i + 2
```

Entrada:
```
6
```
Saída:
```
0
2
4
```

(c)

```
n = int(input())
for i in range(1,n+1):
  if i % 2 == 0:
    print(i)
```

Entrada:
```
6
```
Saída:
```
2
4
6
```

(d)

```
n = int(input())
l = list(range(0,2*n,2))
for i in range(len(l)):
  if i < n:
    print(i)
```

Entrada:
```
6
```
Saída:
```
0
1
2
3
4
5
```

(e)

```
n = int(input())
i = 0
i = n % 2
while i < n:
  print(i)
  i = i + 2
```

Entrada:
```
7
```
Saída:
```
1
3
5
```

6. Qual dos códigos abaixo escreve corretamente a soma dos elementos da variável `lista`?

(a)

```
lista = [2, 5, 8, 10]
s = 0
for x in lista:
  s = s + lista[x]
print(s)
```

Saída:
```
IndexError: list
index out of range
```

(b)

```
lista = [2, 5, 8, 10]
n = len(lista)
s = 0
for x in range(n):
  s = s + x
print(s)
```

Saída:
```
6
```

(c) **Correta**

```
lista = [2, 5, 8, 10]
n = len(lista)
s = 0
for x in range(n):
  s = s + lista[x]
print(s)
```

Saída:
```
25
```

(d)

```
lista = [2, 5, 8, 10]
s = 0
for x in lista:
  s = x
print(s)
```

Saída:
```
10
```

(e)

```
lista = [2, 5, 8, 10]
n = len(lista)
s = 0
for x in range(n):
  s = x
print(s)
```

Saída:
```
3
```

7. Qual dos códigos abaixo escreve apenas o valor total da soma de todos os inteiros entre 1 (inclusive) a 10 (inclusive)?

(a) **Correta**

```
i = 1
soma = 0
while i < 11:
  soma = soma + i
  i = i + 1
print(soma)
```

Saída:
```
55
```

(b)

```
i = 1
soma = 0
while i <= 10:
  soma = soma + i
  i = i + 1
print(1 + soma + 10)
```

Saída:
```
66
```

(c)

```
i = 1
soma = 0
while i <= 10:
  soma = soma + i
  i = i + 1
  print(soma)
```

Saída:
```
1
3
6
10
15
21
28
36
45
55
```

(d)

```
i = 1
soma = 0
while i <= 10:
  soma = soma + i
  i = 1
print(soma)
```

Saída:
```
Loop infinito
...
```

(e)

```
i = 1
soma = 0
soma = soma + i
i = i + 1
print(soma)
```

Saída:
```
1
```

8. Analise o código abaixo e indique qual das alternativas é válida.

```
01.  t = (0, 1, 2, 3)
02.  print(t)
03.  t.append(4)
04.  t[0] = 1
05.
06.  d = {0:"A", 1:"B", 2:"C"}
07.  d[3] = "D"
08.
09.  for i in t:
10.    print(d[i])
```

(a) A linha 2 está incorreta.

(b) A linha 3 está correta.

(c) A linha 7 está incorreta.

(d) A linha 4 está correta.

(e) Nenhuma das anteriores. **Correta**

Saída:
```
(0, 1, 2, 3)
AttributeError: 'tuple' object
has no attribute 'append'
```

9. Qual código irá escrever `r = 22`?

(a)
```
def func(a, b, c):
  soma = a + b + c

soma = 0
a = 5
b = 7
c = 10
func(a, b, c)
print("r =", soma)
```
Saída:
```
r = 0
```

(b)
```
def func(a, b, c):
  soma = a + b + c

a = 5
b = 7
c = 10
func(a, b, c)
print("r =", soma)
```
Saída:
```
NameError: name
'soma' is not
defined
```

(c) **Correta**
```
def func(a, b, c):
  return a + b + c

a = 5
b = 7
c = 10
r = func(a, b, c)
print("r =", r)
```
Saída:
```
r = 22
```

(d)
```
i = 12
a = 10
for j in range(100):
  a += j
r = a + i
print("r =", r)
```
Saída:
```
r = 4972
```

(e)
```
def func():
  if "z" in globals():
    return -1
  else:
    return 22

z = 0
r = func()
print("r =", r)
```
Saída:
```
r = -1
```

Dica: A expressão `"z" in globals()` será `True` se `z` estiver definida no escopo global ou `False` caso contrário.

10. Qual(is) dos códigos abaixo apresentará(ão) erro ao ser(em) executado(s)?

I

```
t = (1, 2, 3)
t.append(4)
print(t)
```

Saída:
```
AttributeError:
'tuple' object
has no attribute
'append'
```

II

```
t = (4, 2, 6)
t[1] = 5
print(t)
```

Saída:
```
TypeError:
'tuple' object
does not support
item assignment
```

III

```
d = {"A":  1, "B":  2}
d["C"] = 3
print(d)
```

Saída:
```
'A': 1, 'B': 2, 'C': 3
```

(a) Apenas I

(b) Apenas II

(c) Apenas III

(d) I e II **Correta**

(e) Nenhuma das anteriores.

11. O que podemos afirmar sobre os códigos abaixo?

I

```
def anexa(t):
  n = len(t)
  for i in range(n):
    t.append(i)
  return t

x = (2, 3, 4)
y = anexa(x)
print(y)
```

Saída:
```
AttributeError:
'tuple' object
has no attribute
'append'
```

II

```
def zera(t):
  n = len(t)
  for i in range(n):
    t[i] = 0
  return t

a = (1, 2, 4, 8)
b = zera(a)
print(b)
```

Saída:
```
TypeError:  'tuple'
object does not support
item assignment
```

III

```
d_ln = {"A":1, "B":2}
d_nl = {1:"A", 2:"B"}

d_ln["C"] = 3
d_nl[3] = "C"

for i in range(1,4):
  print(d_ln[d_nl[i]])
```

Saída:
```
1
2
3
```

(a) III irá executar sem erros. I e II apresentarão erros de execução. **Correta**

(b) I irá executar sem erros.

(c) II irá executar sem erros.

(d) III apresentará erros de execução.

(e) Nenhuma das anteriores.

12. Analise o código abaixo e indique qual das alternativas é válida.

```
01.  def calcula(c):
02.    coringa = 17
03.    resultado = coringa + c + g
04.    i = 0
05.    for i in [1, 2, 3, 4]:
06.      resultado = resultado + 1
07.    resultado = resultado + i
08.    print(resultado)
09.
10.  g = 10
11.  b = 30
12.  c = 40
13.  calcula(b)
```

(a) Será escrito 61.

(b) Será escrito 65. **Correta**

(c) Ocorrerá um erro na linha 3: variável `g` não foi definida.

(d) A execução de uma linha 14 com código `print(coringa)` escreveria 17.

(e) Será escrito 75.

Saída:
65

13. Considere o código abaixo e indique qual das alternativas é válida.

```
01.  def calcula(a,c):
02.    teste = a + b
03.    z = 0
04.    for i in [1, 2, 3]:
05.      z = z + 1
06.    return a + c + z
07.
08.  a = 11
09.  b = 220
10.  c = 33
11.  result = calcula(a,b)
12.  print(result)
```

(a) A execução de uma linha 13 com código `print(z)` escreveria 3.

(b) Ocorrerá um erro na linha 2: variável `b` não definida.

(c) Será escrito 47.

(d) Será escrito 231.

(e) Será escrito 234. **Correta**

Saída:
234

14. O que podemos afirmar sobre os códigos abaixo?

I

```
a = 15
b = 30
if a < b:
  print(1)
if b > 40:
  print(2)
if b == 30:
  print(3)
```

Saída:
```
1
3
```

II

```
def func(l):
  x = [1, 2, 3]
  for i in range(3):
    l[i] = l[i] + x[i]

lista = [3, 3, 3]
func(lista)
print(lista)
```

Saída:
```
[4, 5, 6]
```

III

```
a = 10
b = 10
c = 5
if a == b:
  print(1)
else:
  print(2)
elif c < b:
  print(3)
```

Saída:
```
Loop
infinito
...
```

IV

```
a = 0
b = 1
c = 2
if a == 0:
  print(1)
else:
  print(2)
if c < b:
  print(3)
```

Saída:
```
1
```

(a) O código I escreverá apenas `1`.

(b) O código II escreverá `[3, 3, 3]`

(c) O código III irá executar sem erros.

(d) O código IV irá executar sem erros. **Correta**

(e) Nenhuma das anteriores.

15. Qual dos códigos abaixo irá imprimir apenas `r = 10`?

(a) **Correta**

```
a = 3
b = 7
c = 15
if a > b:
  print("r =", a)
if b < c:
  print("r =", a+b)
else:
  print("r =", c)
```

Saída:
```
r = 10
```

(b)

```
a = 50
b = 10
if a > b:
  print("r =", b)
else:
  print("r =", a)
elif a == b:
  print("r =", a-b)
```

Saída:
```
SyntaxError:
invalid syntax
```

(c)

```
def atribui(l, v):
  l[0] = v

lista = [10, 1, 2]
atribui(lista, -1)
print("r =", lista[0])
```

Saída:
```
r = -1
```

(d)

```
a = 10
b = 20
if a < b:
  print("r =", a)
if b > 15:
  print("r =", b)
```

Saída:
```
r = 10
r = 20
```

(e)

```
a = 10
i = 0
while i < 10:
  r = a - i
  print("r =", r)
  i = i - 1
```

Saída:
```
Loop infinito
r = 10
...
```

16. Após a execução de qual código o valor da variável `lista` será `[1, 2, 3]` e o programa irá executar sem erros?

(a) **Correta**

```
def foo():
  x = [1, 1, 1]
  for i in range(3):
    x[i] = x[i] + i
  return x

lista = foo()
```

Teste:
`print(lista)`

Saída:
`[1, 2, 3]`

(b)

```
a = 5
b = 10
if a == 5:
  lista = [1, 2, 3]
else:
  lista = [0, 0, 0]
elif a < b:
  lista = [1, 2, 3]
```

Teste:
`print(lista)`

Saída:
`SyntaxError:`
`invalid syntax`

(c)

```
def func(a):
  a[0] = 10

lista = [1, 2, 3]
func(lista)
```

Teste:
`print(lista)`

Saída:
`[10, 2, 3]`

(d)

```
lista = [0, 0, 0]
for i in
range(len(lista)):
  lista[i] = -1
```

Teste:
`print(lista)`

Saída:
`[-1, -1, -1]`

(e)

```
a = 10
b = 15
if a < b:
  lista = [1, 2, 3]
if b == 15:
  lista = [0, 0, 0]
```

Teste:
`print(lista)`

Saída:
`[0, 0, 0]`