

# CDIA-ES-MA3-C21 (v0.1.0)

Engenharia de Software

Professor Dr. *Italo S. Vega* (italo@pucsp.br)

FACEI



Pontifícia Universidade Católica de São Paulo

Maio de 2022

## Sumário

<b>Apresentação</b>	<b>2</b>
<b>1 PROBLEMA: Atribuição de Parâmetros</b>	<b>3</b>
<b>2 PROBLEMA: Escopo na Lógica</b>	<b>4</b>
<b>3 PROBLEMA: Escopo no Python</b>	<b>5</b>
<b>4 PROBLEMA: Recursão e Composição</b>	<b>7</b>
<b>5 PROBLEMA: Recursão e Composição em Python</b>	<b>8</b>
<b>Referências</b>	<b>8</b>

## **Apresentação**

Nesta atividade será exercitado o conhecimento de Engenharia de Software desenvolvido ao longo dos encontros.

**Pontuação** Respostas assinaladas com “Não sei” receberão 4 pontos. Caso erre a resposta, a pontuação será zero. Caso acerte a resposta, a pontuação será 10. O total de pontos obtidos nesta avaliação será linearmente normalizado para a escala entre 0 e 10. Faz parte da avaliação a correta interpretação das questões.

# 1 PROBLEMA: Atribuição de Parâmetros

*Contexto* A linguagem Python suporta duas maneiras de vincular os parâmetros de uma função aos seus respectivos valores (inspirado em Guttag (2013), p. 36).



— Considere a seguinte função:

$$mostrarNome \triangleq \lambda p, s : seq\ Char; r : Boolean$$

• **if**  $r$  **then**  $(s \frown \langle , \rangle \frown p)$  **else**  $(p \frown s)$

Espec produz o código Python para implementar a função *mostrarNome*:

```
# MÉTODO
def mostrarNome(p, s, r) :
    if r:
        return s + ', ' + p
    else:
        return p + s
```

Fubã fica em dúvida a respeito das seguintes afirmações:

- I) `mostrarNome('Fubã', 'Fuinha', True) == 'Fuinha, Fubã'`
- II) `mostrarNome('Espec', 'Chair', False) == 'Espec Chair'`
- III) `mostrarNome(r = True, s = 'Espec', p = 'Chair') == 'Espec, Chair'`

*Enunciado* Assinale a alternativa contendo apenas afirmações verdadeiras:

1. I e II.
2. I e III.
3. II e III.
4. I, II e III.
5. “Não sei”.

## 2 PROBLEMA: Escopo na Lógica

*Contexto* Fubã estava com dificuldade para entender melhor a noção de escopo de uma variável (ou espaço de nomes de uma função, predicado ou ação).

VARIABLES

$x : \text{Int}$

$y : \text{Int}$

$z : \text{Int}$

$$\begin{aligned} \text{início} &\triangleq \wedge x = 3 \\ &\wedge y = 2 \\ &\wedge z = 1 \end{aligned}$$

$$\begin{aligned} f &\triangleq \wedge y : \text{Int} \\ &\wedge y = 1 \\ &\wedge x' = x + y \end{aligned}$$

$$\begin{aligned} g &\triangleq \wedge f \\ &\wedge z' = x' \end{aligned}$$



— Se eu considerar o predicado *início* verdadeiro, posso afirmar que:

I)  $x = 3 \wedge y = 2$

Em seguida, assumindo a ação *g* verdadeira:

II)  $x = 4 \wedge y = 2$

III)  $z = 4$

Fubã se lembra que o valor de uma variável é preservado de estado para estado, caso nenhuma ação verdadeira modifique o seu valor.

*Enunciado* Assinale a alternativa contendo apenas afirmações verdadeiras:

1. I e II.
2. I e III.
3. II e III.
4. I, II e III.
5. “Não sei”.

### 3 PROBLEMA: Escopo no Python

*Contexto* Fubã reconsidera a lógica para estudo do escopo de variáveis:

VARIABLES

$x : \text{Int}$

$y : \text{Int}$

$z : \text{Int}$

$\text{início} \triangleq \wedge x = 3$   
 $\wedge y = 2$   
 $\wedge z = 1$

$f \triangleq \wedge y : \text{Int}$   
 $\wedge y = 1$   
 $\wedge x' = x + y$

$g \triangleq \wedge f$   
 $\wedge z' = x'$



— Posso implementar esta lógica em Python:

```
# CENÁRIO de escopo no Python
# VARIABLES
x = 3
y = 2
z = 1
```

```
def f() : # ação "f"
    global x
    y = 1
    x = x + y
```

```
def g() : # ação "g"
    global z
    f
    z = x
```

```
print (x == 1) # <-- (A)
print (y == 2)
print (z == 3) # <-- (B)
```

```
g()  
print (x == 1) # <-- (C)  
print (y == 2)  
print (z == 4) # <-- (D)
```

*Enunciado* Assinale a alternativa com uma afirmação verdadeira:

1. (A).
2. (B).
3. (C).
4. (D).
5. “Não sei”.

## 4 PROBLEMA: Recursão e Composição

*Contexto* Fubã encontrou as seguintes funções (inspirado em Guttag (2013), p. 64):

$$abs \triangleq \lambda x : \text{Int} \bullet \text{if } x > 0 \text{ then } x \text{ else } -x$$

$$m_1 \triangleq \lambda s : \text{seq Int}; f : \text{Int} \rightarrow \text{Int} \mid \#s = 1 \bullet \langle f(\text{head } s) \rangle$$

$$m_2 \triangleq \lambda s : \text{seq Int}; f : \text{Int} \rightarrow \text{Int} \mid \#s > 1 \bullet \langle f(\text{head } s) \rangle \frown m(\text{tail } s, f)$$

$$m \triangleq m_1 \cup m_2$$



— Não sei quais das afirmações são verdadeiras:

- I)  $abs(-5) = 5$ .
- II)  $m(\langle -5 \rangle, abs) = \langle 5 \rangle$ .
- III)  $m(\langle -5, 2 \rangle, abs) = \langle 5, 2 \rangle$ .

*Enunciado* Assinale a alternativa contendo apenas afirmações verdadeiras:

- 1. I e II.
- 2. I e III.
- 3. II e III.
- 4. I, II e III.
- 5. “Não sei”.

## 5 PROBLEMA: Recursão e Composição em Python

*Contexto* Fubã reconsidera as seguintes funções:

$$abs \triangleq \lambda x : \text{Int} \bullet \text{if } x > 0 \text{ then } x \text{ else } -x$$

$$m_1 \triangleq \lambda s : \text{seq Int}; f : \text{Int} \rightarrow \text{Int} \mid \#s = 1 \bullet \langle f(\text{head } s) \rangle$$

$$m_2 \triangleq \lambda s : \text{seq Int}; f : \text{Int} \rightarrow \text{Int} \mid \#s > 1 \bullet \langle f(\text{head } s) \rangle \frown m(\text{tail } s, f)$$

$$m \triangleq m_1 \cup m_2$$



— O código em Python implementa estas funções:

# CENÁRIO de recursão e composição em Python

# Função de estado

```
def abs(x) :  
    return x if x >= 0 else -x
```

# Função de estado

```
def m(s, f) :  
    if len(s) == 1 :  
        return [f(s[0])]  
    else:  
        head, *tail = s  
        return [f(head)] + m(tail, f)
```

# Afirmações em Python

```
abs(-5) == 5           # <-- (I)  
m([-5], abs) == [5]    # <-- (II)  
m([-5, 2], abs) == [5, 2] # <-- (III)
```

*Enunciado* Assinale a alternativa contendo apenas afirmações verdadeiras:

1. I e II.
2. I e III.
3. II e III.
4. I, II e III.
5. “Não sei”.

## Referências

Guttag, J. V. (2013). *Introduction to Computation and Programming Using Python*. The MIT Press.