**SOFTEX RECIFE – DESENVOLVIMENTO WEB FRONTEND**

ALGOTITMO E ESTRUTURA DE DADOS – DESAFIO 6

**ALUNA:** ANANDA CAROLINE FIDELIS E SILVA

**Instruções do projeto**

Crie uma árvore para cada lista abaixo, adicione um valor nela e remova outro, mas, em pelo menos uma das árvores, a remoção deve ser de um nó com dois filhos.

**Lista1 =** 45,20,30,60,81,97,100,7,8,4

**Lista2 =** 15,6,18,3,7,16,20,4

**Obs.:** Realize essa atividade no WORD ou no Bloco de Notas, suba esse arquivo para algum repositório e compartilhe o link no campo ao lado para que outros desenvolvedores possam analisá-lo.

**Resposta:**

class NodoArvore:

def \_\_init\_\_(self, chave=None, esquerda=None, direita=None):

self.chave = chave

self.esquerda = esquerda

self.direita = direita

def \_\_repr\_\_(self):

return '%s <- %s -> %s' % (self.esquerda and self.esquerda.chave,

self.chave,

self.direita and self.direita.chave)

raiz = NodoArvore(45)

raiz.esquerda = NodoArvore(20)

raiz.esquerda.direita = NodoArvore(30)

raiz.direita = NodoArvore(60)

raiz.direita.direita = NodoArvore(81)

raiz.direita.direita.direita = NodoArvore(97)

raiz.direita.direita.direita.direita = NodoArvore(100)

raiz.esquerda.esquerda = NodoArvore(7)

raiz.esquerda.esquerda.direita = NodoArvore(8)

raiz.esquerda.esquerda.esquerda = NodoArvore(4)

print("Árvore 1: ", raiz)

def em\_ordem(raiz):

if not raiz:

return

em\_ordem(raiz.esquerda)

print(raiz.chave),

em\_ordem(raiz.direita)

#----------------------------------------------------------------------

class NodoArvore2:

def \_\_init\_\_(self, chave=None, esquerda=None, direita=None):

self.chave = chave

self.esquerda = esquerda

self.direita = direita

def \_\_repr\_\_(self):

return '%s <- %s -> %s' % (self.esquerda and self.esquerda.chave,

self.chave,

self.direita and self.direita.chave)

raiz2 = NodoArvore2(15)

raiz2.esquerda = NodoArvore2(6)

raiz2.direita = NodoArvore2(18)

raiz2.esquerda.esquerda = NodoArvore2(3)

raiz2.esquerda.direita = NodoArvore2(7)

raiz2.direita.esquerda = NodoArvore2(16)

raiz2.direita.direita = NodoArvore2(20)

raiz2.esquerda.esquerda.direita = NodoArvore2(4)

print("Árvore 2: ", raiz2)

def em\_ordem(raiz2):

if not raiz2:

return

em\_ordem(raiz2.esquerda)

print(raiz2.chave),

em\_ordem(raiz2.direita)