UNIVERSIDADE FEDERAL DE GOIÁS

ENGENHARIA DE SOFTWARE

SOFTWARE CONCORRENTE E DISTRIBUIDO – INF0298

Discente:

Adriel Lenner Vinhal Mori

Docente:

Vagner Jose Do Sacramento Rodrigues

Exercícios sobre concorrência e threads - complementar 1 – 4 jul 2023

Goiânia

2023

Exercícios Threads e Concorrência

1.

```
from threading import Thread


class Node:

    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None

class BinaryTree:

    def __init__(self, root):
        self.root = root

    def traverse(self, node, results):
        if node is None:
            return

        results.append(node.data)

        self.traverse(node.left, results)
        self.traverse(node.right, results)

    def bi_thread_search(self, node, results):
        if node is None:
            return

        thread1 = Thread(target=self.traverse, args=(node.left, results))
        thread2 = Thread(target=self.traverse, args=(node.right, results))

        thread1.start()
        thread2.start()

        thread1.join()
        thread2.join()

    def print_results(self, results):
        for result in results:
            print(result)

if __name__ == "__main__":
    # Criação da árvore binária com o nó raiz contendo "judy".
    tree = BinaryTree(Node("judy"))

    # Construção da árvore binária:
    tree.root.left = Node("bill")
    tree.root.right = Node("mary")

    tree.root.left.left = Node("alcie")
    tree.root.left.right = Node("fred")

    tree.root.right.right = Node("tom")
```

```
    tree.root.left.right.left = Node("dave")
    tree.root.left.right.right = Node("jane")

    tree.root.left.right.left.right = Node("joe")

    # Lista para armazenar os resultados da busca.
    results = []

    # Inicia a busca bi-thread na árvore e preenche a lista de resultados.
    tree.bi_thread_search(tree.root, results)

    # Imprime os resultados da busca.
    tree.print_results(results)
```

Saída:

```
bill
alcie
fred
dave
joe
jane
mary
tom
```

2.

a) F b) V c) V d) F.

3. **E**

S N N N N N N N N N. Está foi a saída mostrada em uma Ide de Java Compiler online, porém na Ide instalada no meu Desktop é apresentada S S S S S S S S S. Ou seja,

No código fornecido, a thread criada está competindo com a thread principal (a thread que executa o método main) pela execução da CPU. Se a thread criada tiver uma oportunidade – processo intermediado pela CPU - de executar antes que a thread principal complete seu loop e imprima os caracteres "S ", então a variável ms será atualizada para "N " e a thread criada imprimirá "N " em vez de "S ". No entanto, se a thread criada não tiver a chance de executar antes que a thread principal termine, então a variável ms permanecerá com o valor inicial "S " e a saída será "S S S S S S S S S ".