Lista 12 Tallys Assis

```
import time
import random
from multiprocessing import Process, Queue

"""
Cada nó envia uma mensagem "election" para todos os outros nós com ID maior que
o seu.

Se um nó receber uma mensagem "election", ele responde com uma mensagem "ok".

Se um nó não receber nenhuma mensagem "election" em um determinado período de
tempo, ele se declara o líder.

Se um nó receber uma mensagem "ok", ele aguarda um determinado período de tempo
para receber mais mensagens "ok".

Se nenhum outro nó responder com uma mensagem "ok", ele se declara o líder.

Se um nó receber uma mensagem "election" de outro nó com ID maior que o seu, ele
envia uma mensagem "ok" e começa o processo novamente a partir do passo 1.

O algoritmo termina quando apenas um nó se declara o líder.
"""

N = 5  # número de nós
WAIT_TIME = 3  # tempo máximo de espera por resposta

class Node(Process):
    def __init__(self, node_id, node_queue, leader_queue):
        super(Node, self).__init__()
        self.node_id = node_id
        self.node_queue = node_queue
        self.leader_queue = leader_queue
        self.leader_id = None

    def run(self):
        self.election()

    def election(self):
        time.sleep(random.randint(1, 5))  # espera aleatória para evitar
conflitos
        print(f"Node {self.node_id} started election")

        # envia mensagem de eleição para nós com ID maior
        for i in range(self.node_id+1, N+1):
            self.node_queue.put(('election', i))

        # aguarda resposta dos outros nós
        responses = []
        start_time = time.time()
        while (time.time() - start_time) < WAIT_TIME and len(responses) < N -
self.node_id - 1:
            if not self.node_queue.empty():
                message = self.node_queue.get()
                if message[0] == 'ok':
                    responses.append(message[1])

        # verifica se foi eleito líder
        if not responses:
            self.leader_id = self.node_id
            self.leader_queue.put(self.node_id)
```

```
            print(f"Node {self.node_id} elected as leader")
        else:
            max_id = max(responses)
            self.node_queue.put(('leader', max_id))
            print(f"Node {self.node_id} forwarded election to node {max_id}")
            # aguarda resposta do líder
            leader_response = None
            start_time = time.time()
            while (time.time() - start_time) < WAIT_TIME and not
leader_response:
                if not self.node_queue.empty():
                    message = self.node_queue.get()
                    if message[0] == 'leader':
                        leader_response = message[1]

            if leader_response == self.node_id:
                self.leader_id = self.node_id
                self.leader_queue.put(self.node_id)
                print(f"Node {self.node_id} elected as leader")
            elif leader_response:
                self.leader_id = leader_response
                self.leader_queue.put(leader_response)
                print(f"Node {self.node_id} acknowledged leader
{leader_response}")
            else:
                print(f"Node {self.node_id} timed out")

if __name__ == '__main__':
    node_queue = Queue()
    leader_queue = Queue()
    nodes = []

    for i in range(1, N+1):
        node = Node(i, node_queue, leader_queue)
        node.start()
        nodes.append(node)

    leader_id = leader_queue.get()
    print(f"Node {leader_id} is the leader")

    for node in nodes:
        node.terminate()
```

```
(venv) tallys@TallysAsus:~/MEGA/Git/PUC/Computacao Distribuida/2023/TP12$ python3 main.py
Node 5 started election
Node 5 elected as leader
Node 5 is the leader
(venv) tallys@TallysAsus:~/MEGA/Git/PUC/Computacao Distribuida/2023/TP12$ python3 main.py
Node 1 started election
Node 5 started election
Node 5 elected as leader
Node 5 is the leader
(venv) tallys@TallysAsus:~/MEGA/Git/PUC/Computacao Distribuida/2023/TP12$ python3 main.py
Node 2 started election
Node 4 started election
Node 4 elected as leader
Node 5 started election
Node 4 is the leader
(venv) tallys@TallysAsus:~/MEGA/Git/PUC/Computacao Distribuida/2023/TP12$
```