Mahdashim
0613

ASSIGNMENT 4

SUZUKI KASAMI

The algorithm for mutual exclusion in a computer network. It uses a token passing mechanism to allow multiple process to access a shared resources in concurrent manner. The algorithm ensure that only one process at a time and has access to the shared resources and prevents other process from entering the critical section

CODE

```
import threading
Class Process
    def__init__(self,id,N,RQ)
        self.id=id
        self.N=N
        self.RQ=RQ

def request_CS(self):
    self.RQ.append(self.id)
    if self.RQ[0]==self.id:
        self.enter_CS()

    else while True:
        if self.RQ[0]==self.d;
            self.enter.CS()
            break

def enter_CS(self):
    Print("Process {self.id} enter critical section")
    self.RQ.P.P(0)
    if len(self.RQ)>0:
        next_process = self.RQ[0]
        self.release_token(nextprocess
```

Date M T W T F S

```
def release_token(self, next_process):
    self.RQ[0] = next_process
    print("Token related to process {next_process}")

Q = []
success = [Process(i, 5, RQ) for i in range(5)]

for process in processes:
    process.request_CS():
```

P20-0613

Date

ASSIGNMENT 5

M T W T F S

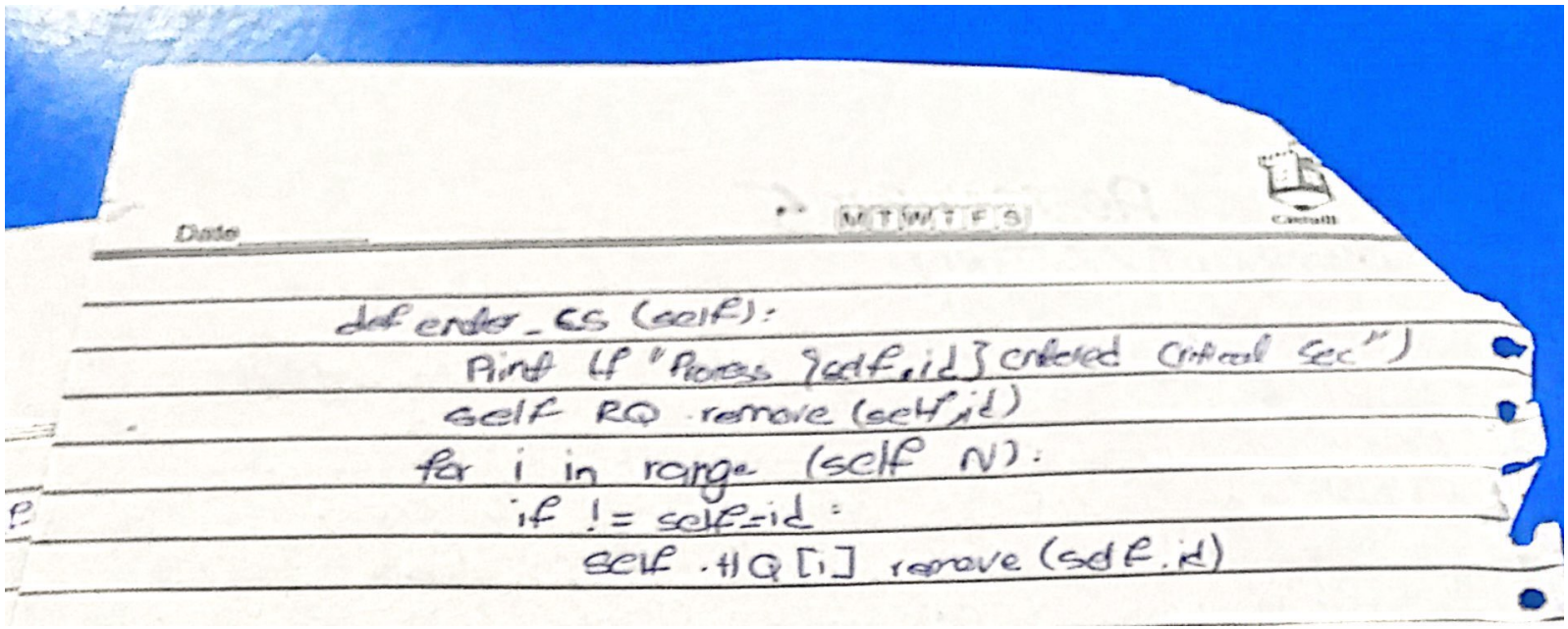
MAEKAWA ALGORITHM

The maekawa algorithm is a distributed mutual exclusion algorithm for synchronizing access to shared resources in a computer network. It provides a mechanism for process to obtain exclusive access to shared resources allowing has to enter critical section and perform some task. The algorithm is based on the concept of a quorum, which is a subset of the process is network

```
import threading
Class Process:
    def __init__(self, id, N, RQ, HQ)
        self.id = id
        self.N = N
        self.RQ = RQ
        self.HQ = HQ

    def request_CS(self)
        self.RQ.append(self.id)
        for i in range(self, N):
            if i != self, id:
                self.HQ[i].append(self, id)

        while len(self.HQ[self.id]) < (self._N-1):
            Pass
        self._enter_CS()
```



```
def enter_CS (self):
    Print (f"Process {self.id} entered Critical Sec")
    self.RQ.remove(self.id)
    for i in range (self.N):
        if i != self.id:
            self.HQ[i].remove(self.id)
```