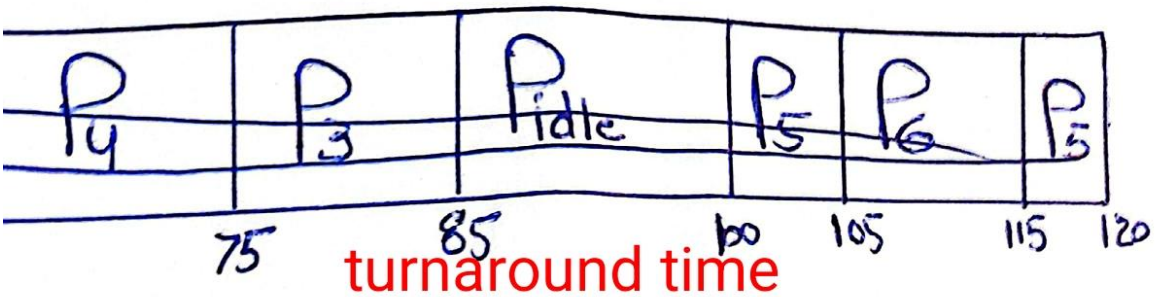


turnaround time

$P_1 = 20 - 0 = 20$

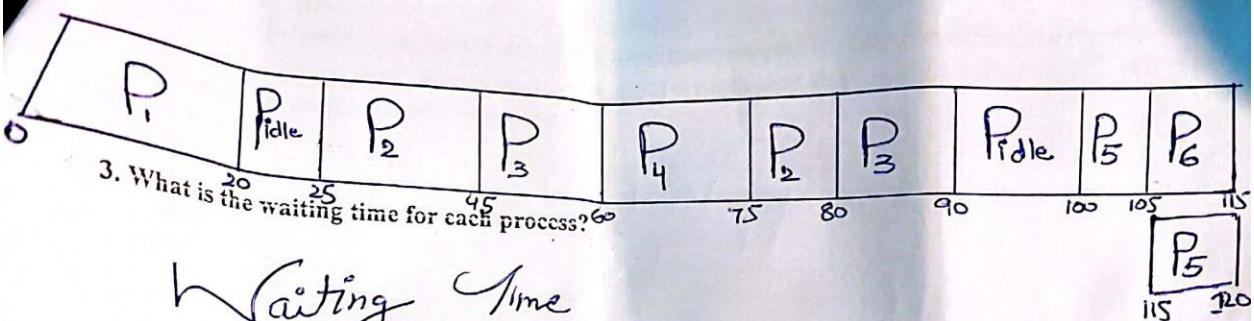
$P_2 = 80 - 25 = 55$

$P_3 = 90 - 30 = 60$

$P_4 = 75 - 60 = 15$

$P_5 = 120 - 100 = 20$

$P_6 = 115 - 105 = 10$



3. What is the waiting time for each process?

Waiting Time

$P_1 = 20 - 0 - 20 = 0$

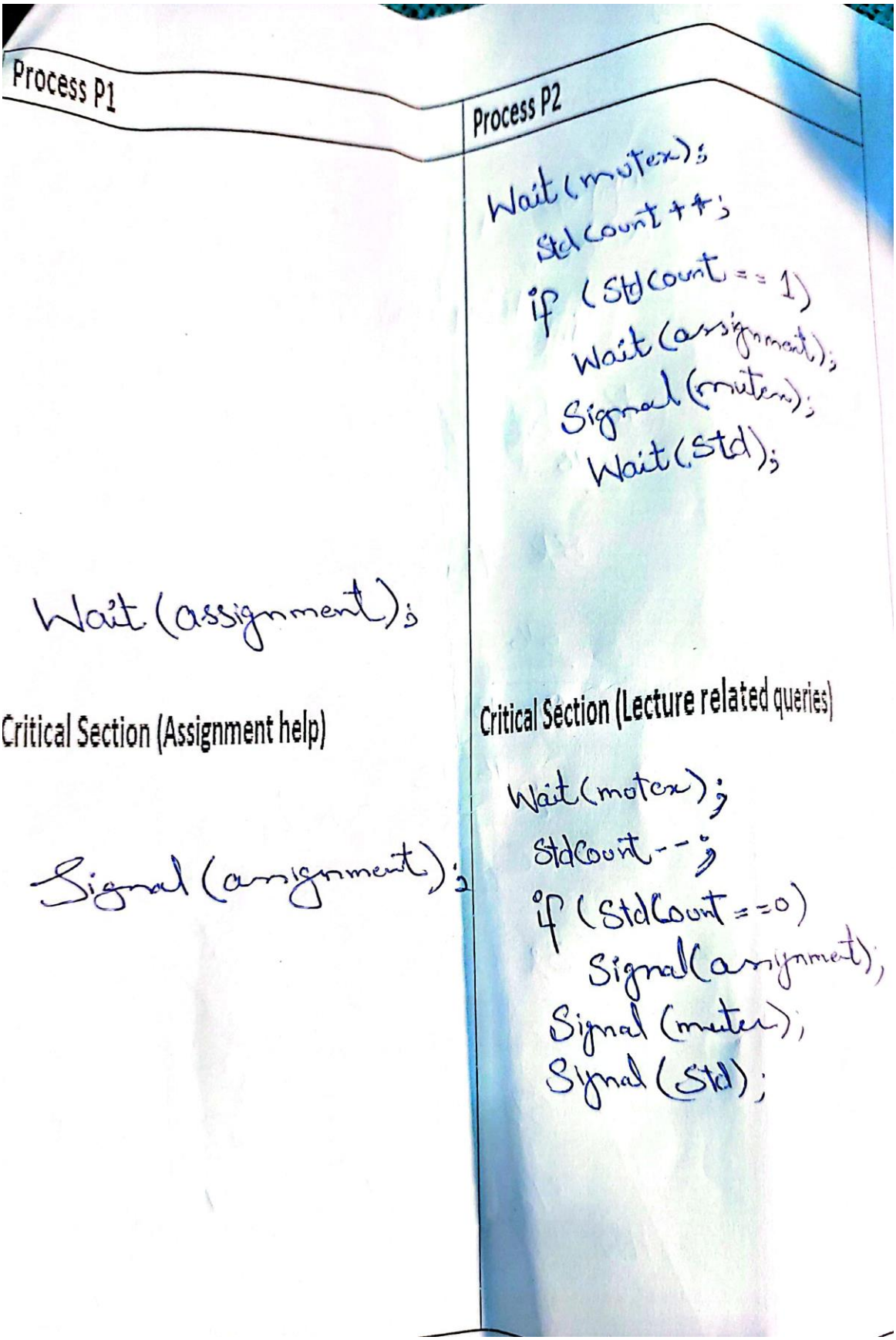
$P_2 = 80 - 25 - 25 = 30$

$P_3 = 90 - 30 - 25 = 35$

$P_4 = 75 - 60 - 15 = 0$

$P_5 = 120 - 100 - 10 = 10$

$P_6 = 115 - 105 - 10 = 0$


Process P1
Process P2
Wait (mutex);
StdCount++;
if (StdCount == 1)
Wait (assignment);
Signal (mutex);
Wait (Std);
Wait (assignment);
Critical Section (Assignment help)
Critical Section (Lecture related queries)
Signal (assignment);
Wait (mutex);
StdCount--;
if (StdCount == 0)
Signal (assignment);
Signal (mutex);
Signal (Std);

Question No. 3

A university computer science department has a teaching assistant (TA) who helps und... students with their programming assignment during regular office hours. TA's responsibilities... helping students in their programming assignments and handling student's lectures related ... So we have two category of students

1: First type of students are those who seek help related to assignment. These types of students cannot ente... TA's room simultaneously (because we want to ensure assignment's confidentiality). We can denote these types of processes with P1.

2: Second type of students are those who have lecture related queries. These types of students can enter TA's room simultaneously. We can denote these types of processes with P2.

The TA's office is rather small and has room for only one desk with three visitor chairs and a computer.

Constraints:

- If two or more P2 processes enter the room simultaneously, then no adverse effect will occur but remember we have only three visitor chairs so we can accommodate only three P2 processes simultaneously.
- On the other hand, If a process P1 is in the room then no other process either P1 or P2 can enter room.
- Furthermore, if P1 and P2 both want to the enter room then we have to give priority to P2.

Your task is to enforce these rules using semaphores. Give your solution in the form of pseudocode.

//declare any shared variables/semaphores here in this block

```
Semaphore assignment = 1;
mutex = 1;
int std count = 0;
CountingSemaphore = std = 3;
```