**1.**

**ANS.**

n! =n factorial = n × n − 1 × n − 2 × … × 2 × 1

**2.**

**ANS.**

**Shortest Job First:**

* Shortest Job First is a scheduling algorithm that assigns the length of each process's next CPU burst/execution time to each process.
* The process is then provided to the CPU with the least amount of CPU burst from the waiting Queue.
* Shortest Job First is optimal in that it delivers the least average waiting time for each process for a given collection of processes and their CPU execution periods.
* It is not practically implementable because the burst-time of a process can not be predicted in advance.
* We may not know the the length of the next CPU burst but expect the next CPU burst will be similar to the length of the previous ones.

**Let, ti  be the actual burst time of ith process and Tn+1 be the predicted Burst time for n+1th process.**

**For the given n processes:**

**Tn+1= 1/n (**Σi=1 to n ti**)**

**Exponential average:**

**Tn+1=** αtn + (1 - α)Tn

* Where α is smoothing factor and 0<= α <= 1.
* **tn is the actual burst time of the nth process.**
* **Tn  is the predicted burst time of the nth process.**

**General term,**

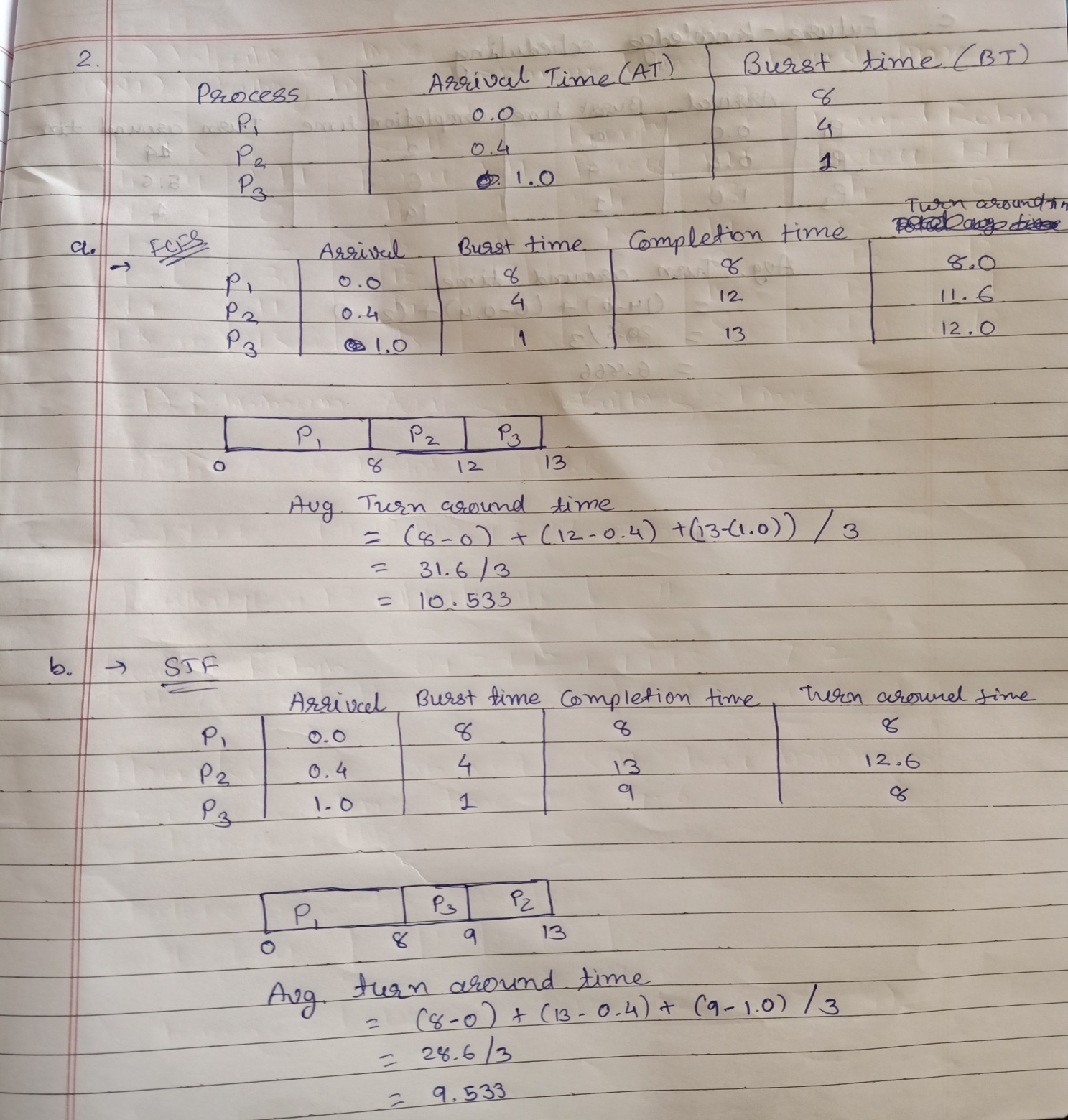
Αtn + (1 - α) αtn-1 + (1 - α)2αtn-2 +…+(1 - α)jαtn-j +…+ (1 - α)n+1T0

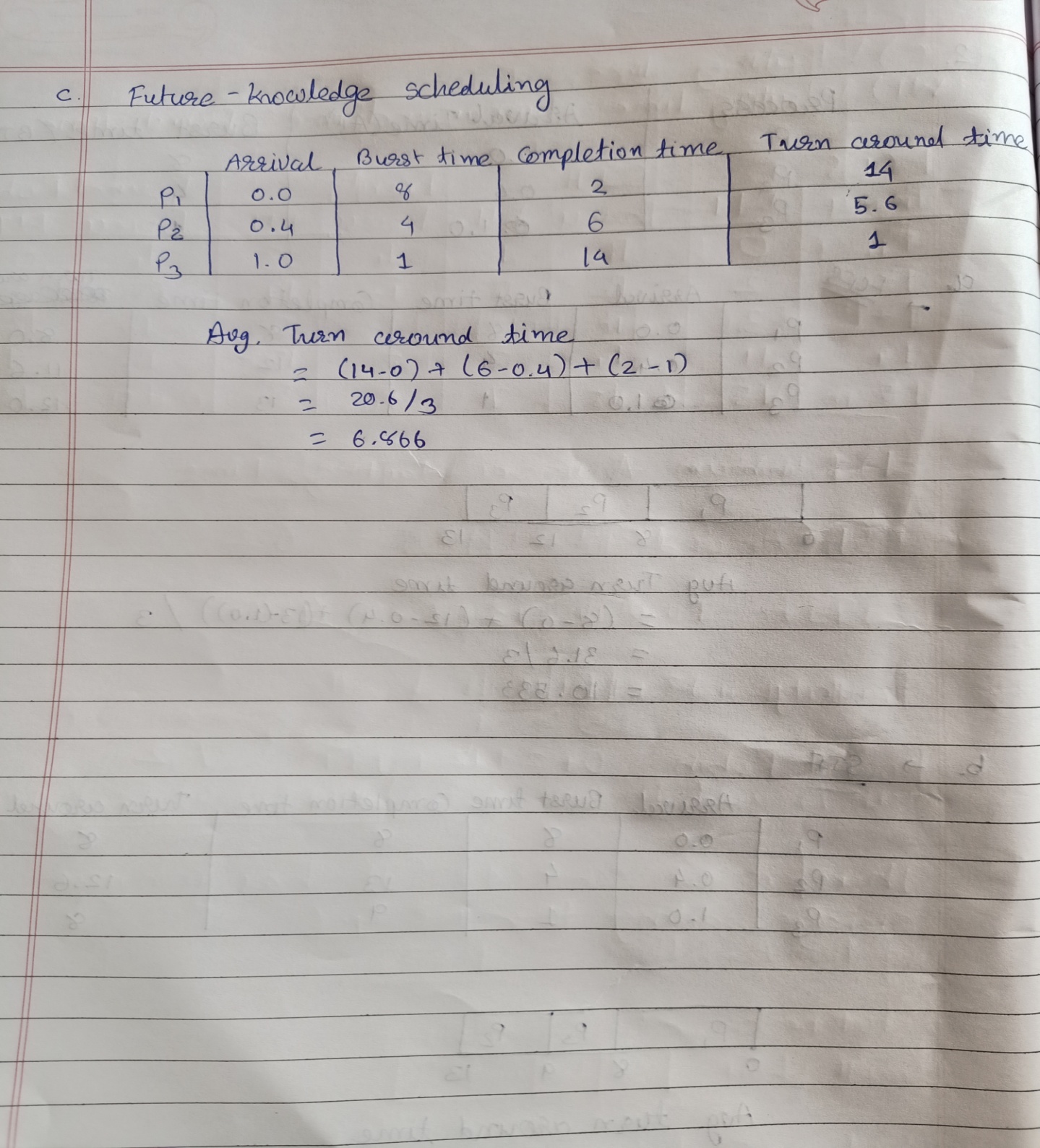
To is the constant or overall system storage.

Repeating the process over and putting shorter jobs in front of longer ones, we will eventually completely order the starting set and achieve the minimal average waiting time.

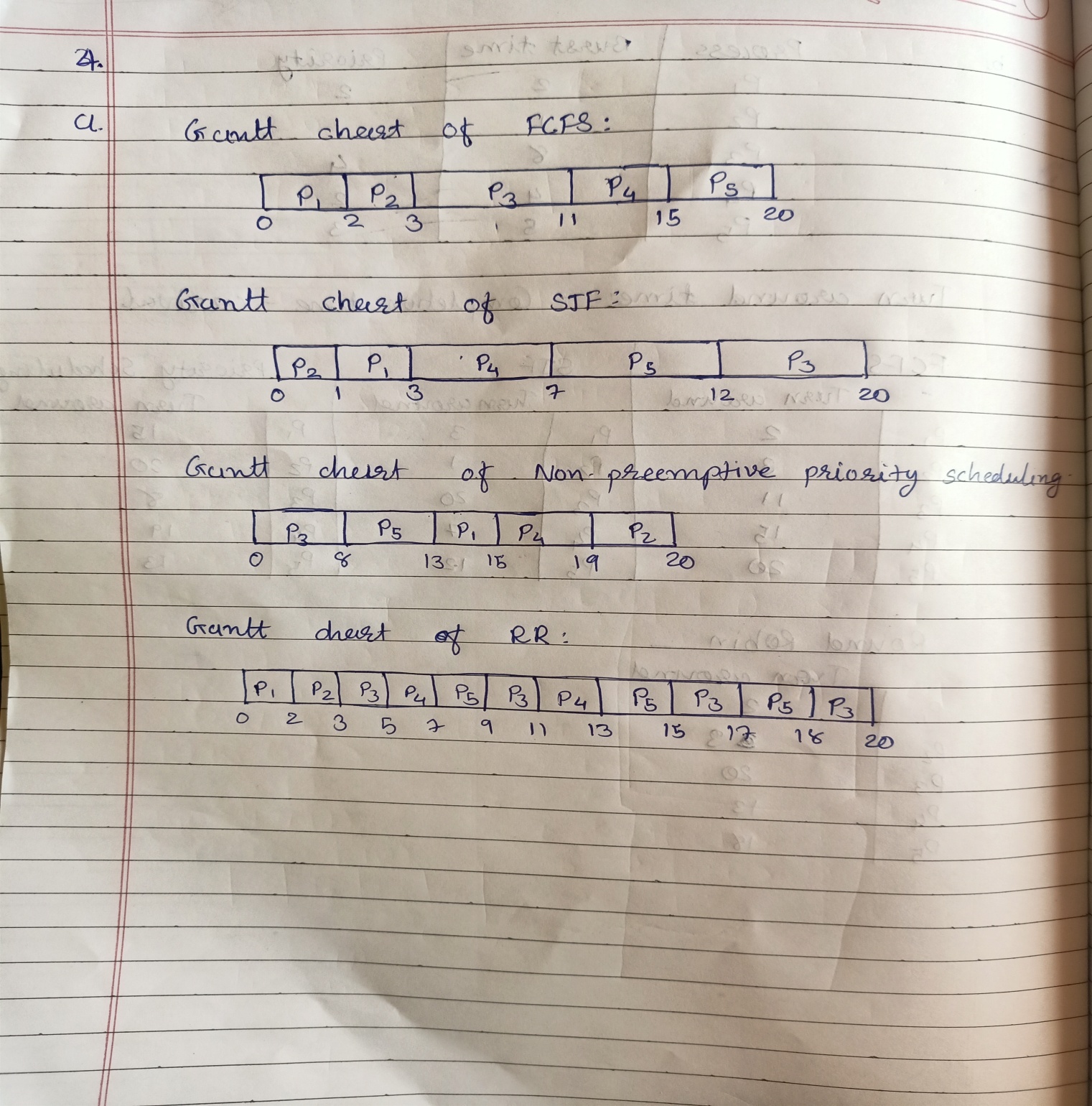
**3.**

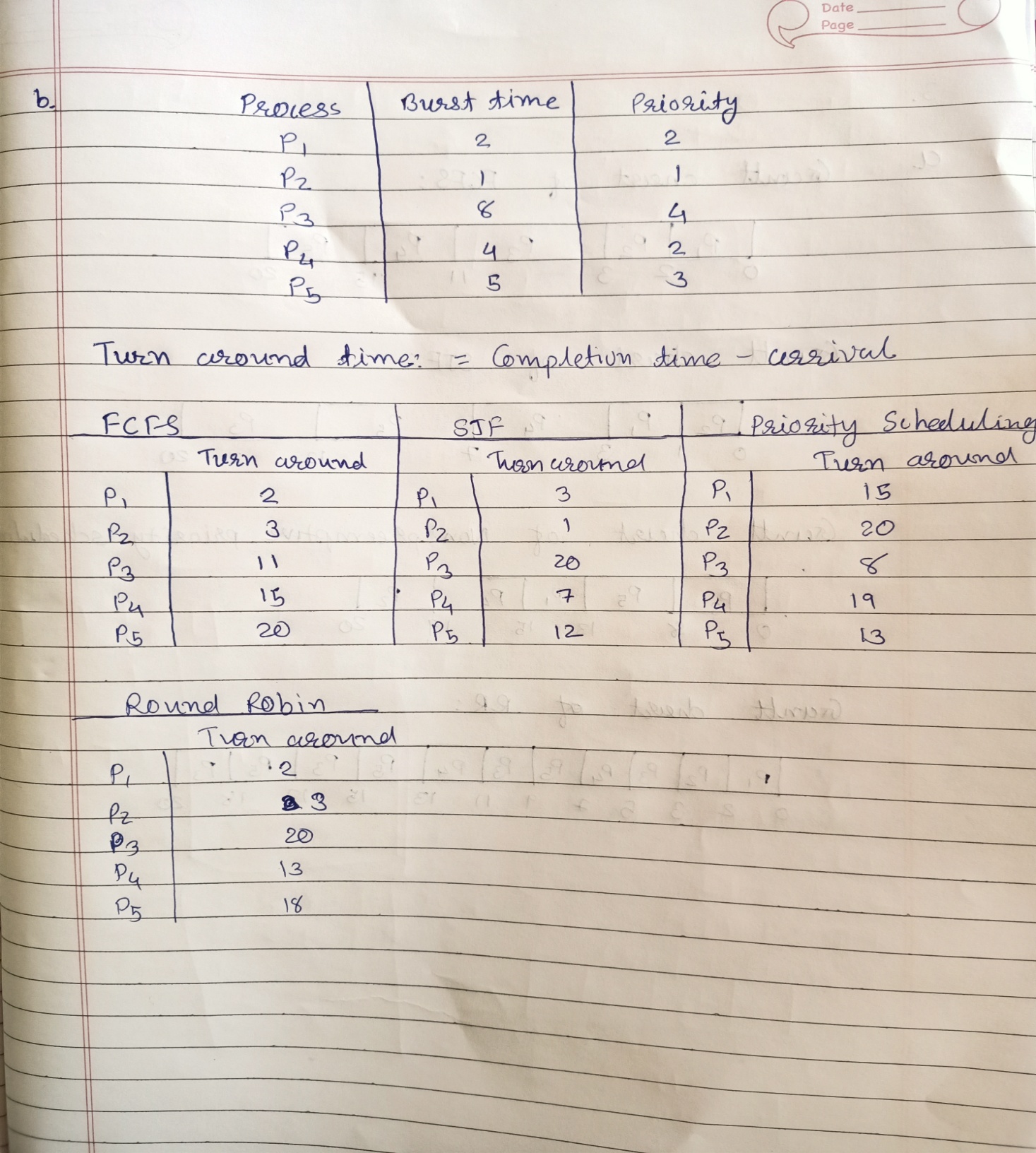
**5.3 ANS.**

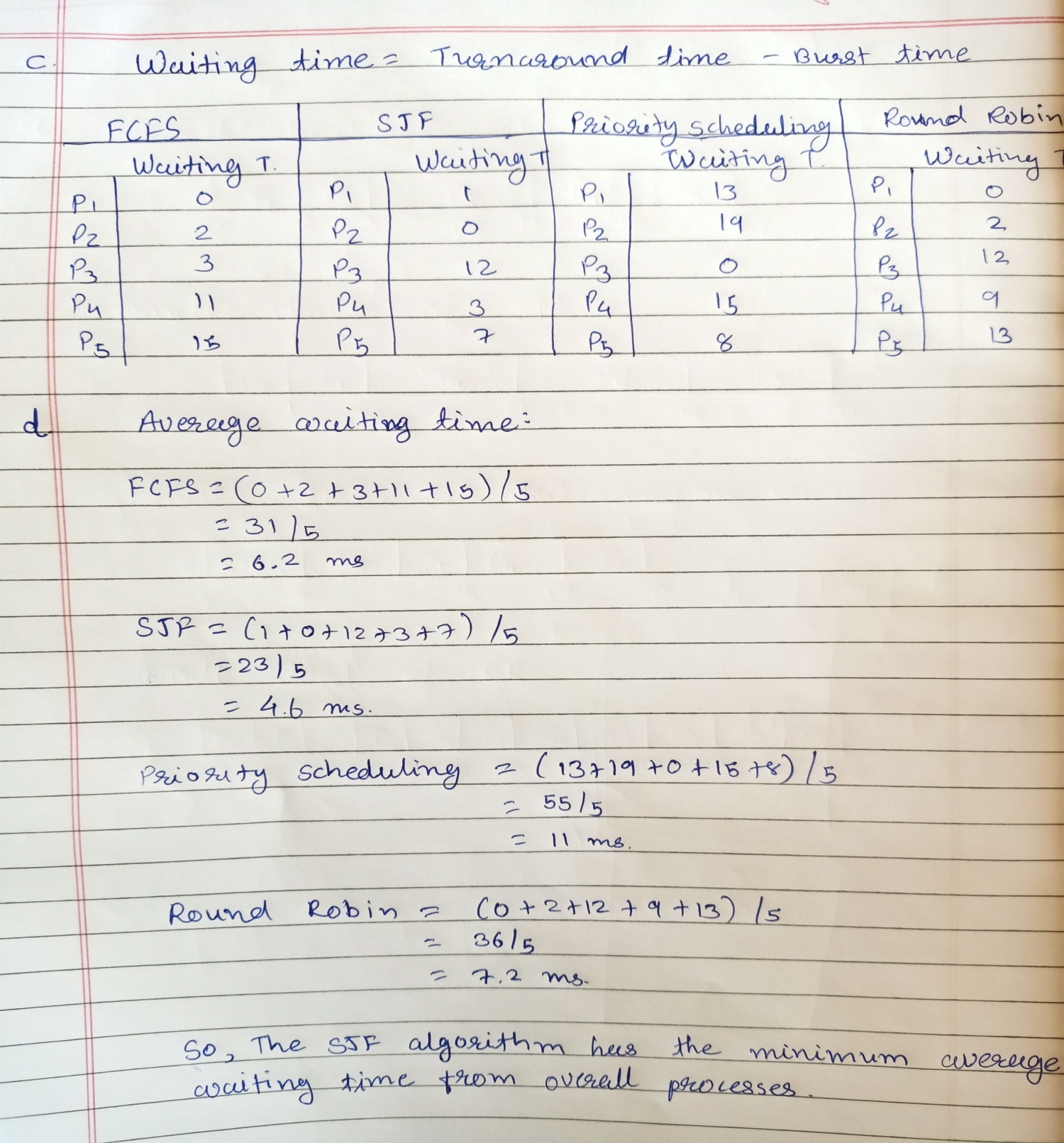
****

****

**5.4 ANS.**

****

****

****

**4.**

**Ans.**

* In the first situation, when the available space is given over to a smaller party if the larger party is unable to seat completely, this is known as starvation in OS (shortest job first) or DBMS and the larger party may be forced to wait indefinitely. It allows smaller parties to make better use of available resources while keeping the larger party waiting for an extended period of time
* If parties are arranged strictly in the order in which they arrive, resources will be wasted, but all parties will have a fair shot. This is an example of the operating system's first come, first served strategy. As a result, the average waiting time for the parties increases, but it ensures that all parties have an equal opportunity to dine.

**5.**

**ANS**.

* Because they spend the majority of their time waiting for I/O operations on some device, our scheduler will prefer I/O bound processes.
* The speed of I/O bound programs is determined by this. As a result, they don't require a lot of CPU time or a lot of short CPU bursts, which means they'll be prioritized by the scheduler.
* I/O bound programs will ultimately terminate, and CPU bound programs will continue to execute, therefore CPU bound processes will not go in starvation indefinitely.
* I/O bound programs are preferred by the scheduler since they spend less time on the CPU. I/O bound programs will eventually finish, therefore CPU bond programs will not go in starvation indefinitely.