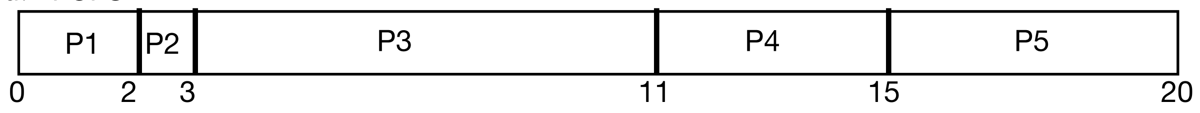
Craig Huff
Homework4
ID #: 009841390

1. Consider the following set of processes, with the length of the CPU burst time given in milliseconds:

Process	BurstTime	Priority
P1	2	3
P2	1	1
P3	8	5
P4	4	2
P5	5	4

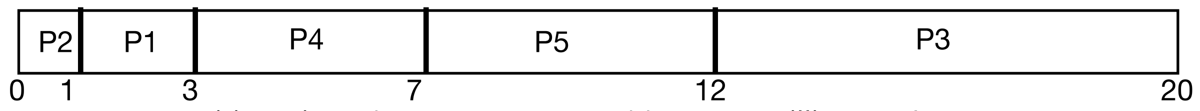
The processes are assumed to have arrived in the order *P*1, *P*2, *P*3, *P*4, *P*5, all at time 0. Use a software to draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms:

a. FCFS



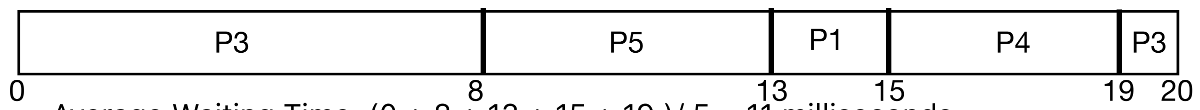
Average Waiting Time: (0 + 2 + 3 + 11 + 15)/ 5 = 6.2 milliseconds

b. nonpreemptive SJF



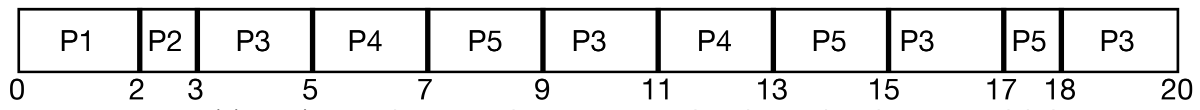
Average Waiting Time: (0 + 1 + 3 + 7 + 12)/ 5 = 4.6 milliseconds

c. nonpreemptive priority (a larger priority number implies a higher priority)



Average Waiting Time: (0 + 8 + 13 + 15 + 19)/ 5 = 11 milliseconds

d. and RR (quantum = 2), and calculate the average waiting time for each algorithm.



Average Waiting Time = (0 + 2 + (3 + 4 + 4 + 1) + (5 + 4) + (7 + 4 + 2)) / 5 = 7.2 milliseconds

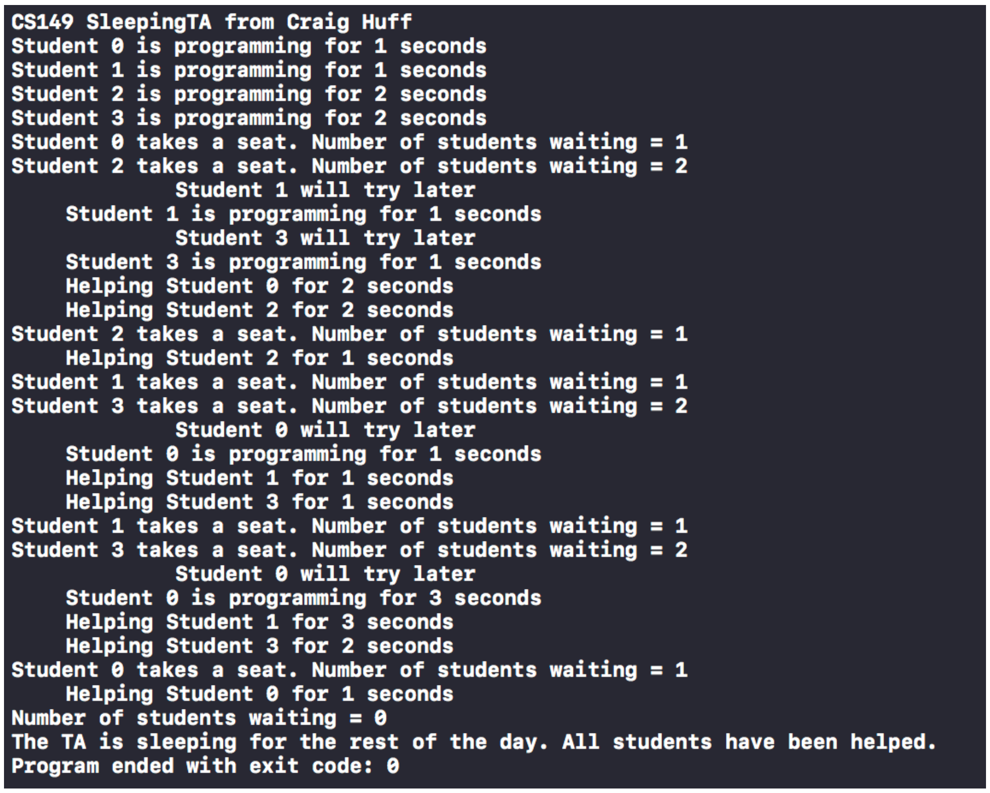
2. Describe the exact calling sequence of two semaphores and mutex lock in TA thread and student thread, respectively.

a. TA Thread
The TA thread will wake up when it is called upon by the student, since it's waiting for the student semaphore to call it. Once it's called, it will acquire the mutex lock and change run through the critical section. Once done in the critical section it will unlock the mutex variable, and let the student know it's

done working.

b. Student Thread
The student thread will acquire the lock so it can check the number of seats available and either add itself to the waiting queue or come back later. If the student is adding itself to the queue, it will complete the critical section, release the mutex lock, and use a semaphore to let the TA know it’s ready for help and then wait for the TA’s semaphore to complete it’s tasks. If the student is coming back later, it will release the mutex lock and print out that it’s coming back later.

Sample Output:



```
CS149 SleepingTA from Craig Huff
Student 0 is programming for 1 seconds
Student 1 is programming for 1 seconds
Student 2 is programming for 2 seconds
Student 3 is programming for 2 seconds
Student 0 takes a seat. Number of students waiting = 1
Student 2 takes a seat. Number of students waiting = 2
             Student 1 will try later
    Student 1 is programming for 1 seconds
             Student 3 will try later
    Student 3 is programming for 1 seconds
    Helping Student 0 for 2 seconds
    Helping Student 2 for 2 seconds
Student 2 takes a seat. Number of students waiting = 1
    Helping Student 2 for 1 seconds
Student 1 takes a seat. Number of students waiting = 1
Student 3 takes a seat. Number of students waiting = 2
             Student 0 will try later
    Student 0 is programming for 1 seconds
    Helping Student 1 for 1 seconds
    Helping Student 3 for 1 seconds
Student 1 takes a seat. Number of students waiting = 1
Student 3 takes a seat. Number of students waiting = 2
             Student 0 will try later
    Student 0 is programming for 3 seconds
    Helping Student 1 for 3 seconds
    Helping Student 3 for 2 seconds
Student 0 takes a seat. Number of students waiting = 1
    Helping Student 0 for 1 seconds
Number of students waiting = 0
The TA is sleeping for the rest of the day. All students have been helped.
Program ended with exit code: 0
```