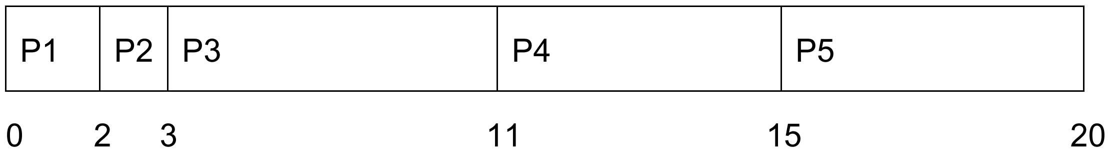
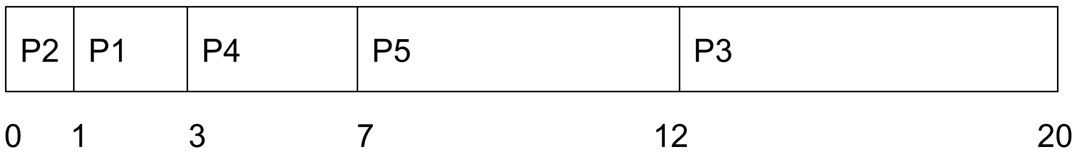
Q1: (1) FCFS:



P1: 0 P2: 2 P3: 3 P4: 11 P5: 15

Average waiting time: (0 + 2 + 3 + 11 + 15) / 5 = 31/5 = 6.2

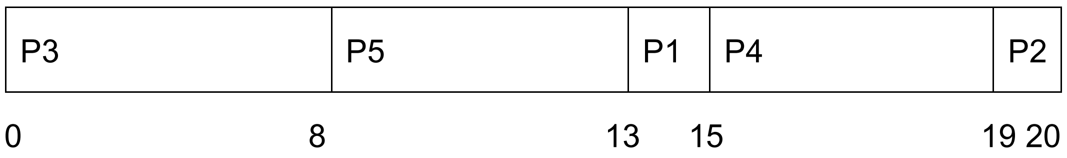
(1) Nonpreemptive SJF



P2: 0 P1: 1 P4: 3 P5: 7 P3: 12

Average waiting time: (1 + 3 + 7 + 12) / 5 = 23 / 5 = 4.6

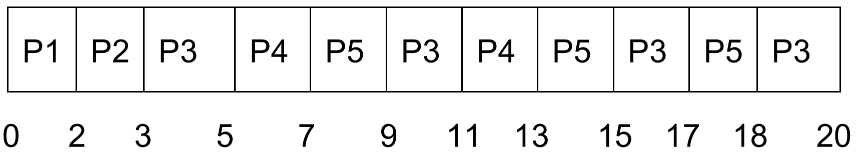
(2) Nonpreemptive priority



P3: 0 P5: 8 P1: 13 P4: 15 P2: 19

Average waiting time: (0 + 8 + 13 + 15 + 19) / 5 = 55 / 5 = 11

(3) RR (quantum = 2)



Average waiting time:

P1: 0 P2: 2 P3: 3 + (9 - 5) + (15 - 11) + (18 - 17) = 12

P4: 5 + (11 - 7) = 9 P5: 7 + (13 - 9) + (17 - 15) = 13

Average waiting time: (0 + 2 + 12 + 9 + 13) / 5 = 7.2

Q2:

The mutex_lock is to protect global variable waiting_students. The semaphore students_sem and ta_sem are used to control sequence of execution.

Student thread:	In TA thread:
1. Programming 1-3 seconds	1 sem_wait(&students_sem): Wait for students to appear
2. Lock mutex_lock to check value of waiting_students: (1) If waiting_students >= 2: Unlock mutex_lock Print try later and go to 1. (2) if waiting_students < 2: waiting_students++; print taks a seat; unlock mutex_lock	2 lock mutex_lock Print help a student, update waiting_students Unlock mutex_lock
3. sem_post(&students_sem): notify ta students arrived	3 sem_post(&ta_sem): notify a student that he receives help now
4 sem_wait(&ta_sem): wait for ta to start help me print receive help	4 lock waiting_students to check if it equals 0, if yes, go to 1 to wait for new students to appear. Otherwise, go to 2 to help waiting students sitting in the hall way.