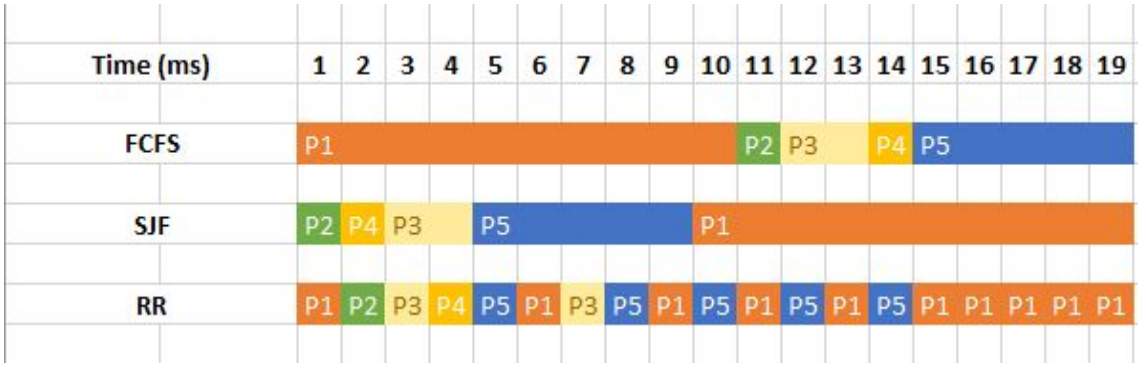
1. The goal of the scheduler is to ensure the CPU is as busy as possible ie no time is wasted. Distinguishing between the two kinds of processes is the helps determines which processes to give priority to. For example, assuming I/O processes are often shorter than CPU bound, that means they're more likely to pause and wait for input before finishing. During that wait time, a CPU bound process could be executed and completed thus saving time.
2.
    a.



    b.

Algo\Process	P1	P2	P3	P4	P5
FCFS	10	11	13	15	19
SJF	19	2	4	2	9
RR	19	2	7	4	14

    c.

    d.

Process	1	2	3	4	5	Average
FCFS	0 ms	10 ms	11 ms	13 ms	14 ms	5 ms
SJF	9 ms	0 ms	2 ms	1 ms	4 ms	3.2 ms
RR	9 ms	1 ms	5 ms	3 ms	9 ms	5.4 ms

    e. Lowest average wait time is the SJF algorithm

3. Unless the system has hyperthreading available, both multiprocessor and single processor systems will have the same performance since threading, although seems parallel, is just sequential processing at an virtually instant rate.
However, the multi-core system would perform faster than the single-processor if hyperthreading is available to allow for actual parallel processing.

4.

a. Operating systems. Background process like the print spooler are always running to detect when a print job is released. Microsoft Windows

b. GUI Apps need multithreading to maintain interactive while they're processing input at the same time. Example would be a word processor. It has one process checking grammar and spelling while still accepting input from the user

c. Image/Video rendering apps. In order to render graphics faster, the work is split into sections and independently rendered by multiple threads which are then pieced back together. Eg: Photoshop

5.

a. In the case of the entry section, at most one thread from the pool is allowed into the critical section at a time but not allowed to execute unless the **mustWait** condition is false. This ensures every thread having at least some chance to get into the entry section allowing for **lockout-freedom**. But before the thread gets a pass into **active** or is placed in **waiting**, the state of the other shared variables are updated so future threads have the most recent information. This thread also releases the section appropriately to prevent **deadlocks** and starvation of other threads. This is the first level/layer critical section (entry level) before the second section where the thread is allowed to actually execute its instructions.
   At the second level, the 3 or less threads in **active** also enter the critical section however, the **mutex** is used to prevent modifying shared resources all at once. First, the section is locked off by (assuming the soonest) a thread in the n group. This thread in this area must update all the shared variables /or not if active != 0 before it releases the section to the next thread. The **mustWait** variable works together with the active variable to ensure the active group is empty before another group of 3 is created.
   In both sections, use of the **mutex** encourages **mutual exclusion** in them allowing for more reliable updates to the shared resource.

b. During the block release in the second critical stage, a new thread may have arrived before an unblocked thread has a chance to resume operation and may fall behind the new arrival if there's still room in the active slots.

c. This pattern of using semaphores such that process that have been blocked at the entry stage are unblocked by those that have completed the critical section and are just about leaving. In this way, the unblocking process does the unblocking for the blocked (hence I'll do it for you) so that the process doesn't have to unlock itself and go through the entry section again. The goal of this is to minimize cutting in line by other processes.

6.

Assessment

1. No. Did not have the time to review all of my work.

2.
    a. Visually assessing code still proves difficult
    b. Some questions are still either too abstract or too open to even know how to start answering. For instance questions 4 and also pretty much all of question 6. They feel “directed” to coerce a particular answer

3.

October 12	Did problems 1 through to 3	Approx. 3	Made good progress and it was a chance to read up on the material and get a better understanding
October 20	Decided to skip question 5. Did for and 6 and started working on code	Approx 7 hours	Answering question a lot of time as I wasn’t really sure what was meant by “why it’s correct”. The programming also took up the rest of the time and I realised I wouldn't be able to complete the entire assignment. I couldn’t devote time to it because of midterms and other courses which has closer deadlines