1. I/O bound require voluntary context switches where CPU bound require non-voluntary context switches. Because the voluntary context witches occur when process has given up control of the CPU and the non-voluntary context switches occurs when the CPI has taken away from a process.
2. For the FIFO:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task | Length | Arrival Time | Turnaround | Waiting time |
| 0 | 85 | 0 | 85 | 0 |
| 1 | 30 | 10 | 115 | 85 |
| 2 | 35 | 15 | 150 | 115 |
| 3 | 20 | 80 | 170 | 150 |
| 4 | 50 | 85 | 220 | 170 |

Average turnaround time = 148

For the SJF (assume non-preemptive)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task | Length | Arrival Time | Turnaround | Waiting time |
| 0 | 85 | 0 | 85 | 0 |
| 1 | 30 | 10 | 135 | 105 |
| 2 | 35 | 15 | 170 | 135 |
| 3 | 20 | 80 | 105 | 85 |
| 4 | 50 | 85 | 220 | 170 |

Average turnaround time = 143

For RR

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task | Length | Arrival Time | Turnaround | Waiting time |
| 0 | 85 | 0 | 220 | 140 |
| 1 | 30 | 10 | 80 | 40 |
| 2 | 35 | 15 | 125 | 70 |
| 3 | 20 | 80 | 145 | 30 |
| 4 | 50 | 85 | 215 | 70 |

Average turnaround time = 157

1. three requirements to the critical section:

a. Mutual exclusion (mutex)

if a process is executing in its critical section, then no other processes can be executing in their critical sections

b. Progress

if some process is not in the critical section, it cannot prevent some others from entering the critical section

a process in the critical section will eventually leave it

c. Bounded waiting (no starvation)

if some process is waiting on the critical section, it will eventually enter the critical section

1. if only use turn to determine the critical-section is available or not is not enough, we should add a bool flag like this:

while(flag[j] && turn == j)

1. the deadlock will happen if all the philosophers grab the right chopstick at the same time, then they will all waiting for the available left chopstick. The solution for this problem would be even number person grab the chopstick from their left, odd number grab the chopstick from their right.
2. True
3. Since the semaphores are integer variables that are used to solve the critical section problem and if it is negative or zero, then no operation is performed. So, if the wait operation is not atomic, it may result in two wait operations are executed on a semaphore at the same time when S is just 1, that violating mutual exclusion.
4. False, because spinlocks are not appropriate for single processor systems since there is only one cpu processor to do the job and no extra process can get the waiting process out.
5. Test\_and\_set(), compare\_and\_swap()
6. Suppose the account has 500, if withdraw 100 and deposit 100 at the same time, the account’s balance could be from 400 to 600 (depends on who save the balance lastly).

The solution for this problem is using a lock to guarantee that only one person can operate the account at a time.