1.

There are two processes P_0 and P_1.

At time T_0

- In=false
- P_0 is scheduled and P_0 try to go to critical section.
- P_0 read In = false, then time out. P_0 status change from running state to ready state.

At Time T_1

- P_1 is scheduled. P_1 try to enter the critical section.
- P_1 read In =false and set In = ture then go to critical section.

At Time T_2

- Sometimes later in the critical section, P_1 time out. P_1 status changes from running to ready state.
- P_0 rescheduled, and try to enter critical section. P_0 already read In =false before, P_0 set In = true again and enter the critical section.
- Now P_0 and P_1 are in critical section (violate mutual exclusion condition)

2.

- When thread makes a blocking system call, the entire process will be blocked. Only one thread can access the Kernel at a time, so multiple threads are unable to run in parallel on multiprocessors.

3.

Shortest remaining time first (preemptive):

P_1	P_2	P_3	P_5	P_2	P_4	P_1
3	5	9	12	17	24	32

- Average waiting time – ((24-3)+(12-5)+0+(17-7)+(9-8))/5 =(21 + 7 + 0 + 10 + 1)/5 = 7.8
- Average turnaround time – ((32-0)+(17-3)+(9-5)+(24-7)+(12-8)/5=(32 + 14 + 4+ 17 + 4)/5 = 14.2

Preemptive priority queue:

P1	P2	P3	P4	P2	P1	P5
3	5	9	16	21	29	32

- Average waiting time – ((21-3)+(16-5)+0+(9-7)+(29-8))= (18+11+ 0+ 2+ 21)/5 =10.4

- Average turnaround time – ((29-0)+(21-3)+(9-5)+(16-7)+(32-8)=
  (29+ 18+ 4+ 9+ 24)/5 = 16.8

4.

- No two processes may be simultaneously inside their critical regions – mutual exclusion
- No process running outside its critical region may block other processes
- No process should have to wait forever to enter critical region
- No assumptions may be made about speeds or the number of CPUs.

5.

Sol) let's assume : empty = 0, full = N, mutex =1 at time T

- Producer is scheduled : produce item ,down mutex (now mutex =0), try to down empty. Since empty =0, producer cannot finish down operation and sleep on semaphore empty.
- Consumer is scheduled: down full (now full = N-1), then try to down mutex. Since mutex is already down by producer, consumer cannot finish down operation and sleep on semaphore mutex.
- Now producer and consumer sleep forever!

6.

- A Web server that services each request in a separate thread.
- A parallelized application such as matrix multiplication where different parts of the matrix may be worked on in parallel.

7.

- User-level threads are unknown by the kernel, whereas the kernel is aware of kernel threads.
- Kernel threads need not be associated with a process whereas every user thread belongs to a process.
- Kernel threads are generally more expensive to maintain than user threads as they must be represented with a kernel data structure.

8.

Sol) Lets assume a short-term scheduler use the priority to select a process from the ready queue. At time t_0 , there is only one process P_L with low priority in the ready queue. The short term scheduler select P_L and let it use CPU. Then P_L enter a critical region (section). At time t_1, a process P_H with higher priority becomes ready state. The short-term scheduler stop P_L to use CPU. Now P_H and P_L are in ready queue. The short-term scheduler select higher priority process P_H and let it use CPU. P_H try to get into the critical section. P_H must wait outside critical section since P_L is already in the critical section. Since P_L has lower priority, P_L never get change to use CPU. P_H never be able to enter critical session.