1.

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

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.

- Since kernel only involved in creation of a shared memory, to access shared memory does not need context switch between kernel and process.

4.

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.

5.

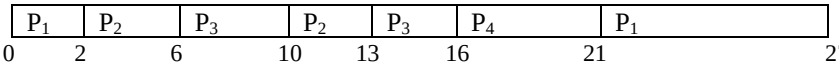
Let's assume Permit =0 at time T0
P0 tries to enter C.S. and can enter since Permit =0.
P0 finish its job in C.S. and set Permit =1
P1 is currently running outside C.S ,it is terminated with fatal error.
P0 tries to enter C.S. again but P0 never can.

6.

Let's assume at time T_0: empty = N, full = 0, mutex =1

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

7.



Average Waiting time = ((21 – 2) + (10 – 6) + ((6 – 4) + (13 – 10)) + (16 – 6))/4 = 19+4+5+10/4= 9.5

Average Turnaround time = ((27 – 0) + (13 – 2) + (16 – 4) + (21 – 6))/4 =27+11+12+15/4= 16.25

8.

- Running state – a process is using CPU for a calculation
- Ready state – a process is waiting for CPU (short-term scheduler)
- Blocked state – a process is waiting for I/O finish
- Transaction 1 – a process need I/O
- Transaction 2 – a process time out
- Transaction 3 – short-term scheduler select a process to run
- Transaction 4 – a process finish I/O and ready to run