1. (1 pt.)

   (Answer)

   - Text section – executable code
   - Data section – Global variable and Static variables
   - Heap – space for dynamic memory allocation
   - Stack section – store local variables when a function all

2. (1 pt.)

   (Answer)

   Let's assume a short-term scheduler uses preemptive priority to select a process from the ready queue. At time t_0, there is only one process, PL, with low priority in the ready queue. The short-term scheduler selects PL and allows it to use the CPU. Then, PL enters a critical section. At time t_1 , a process PH with higher priority becomes ready. The short-term scheduler stops PL from using the CPU. Now both PH and PL are in the ready queue.

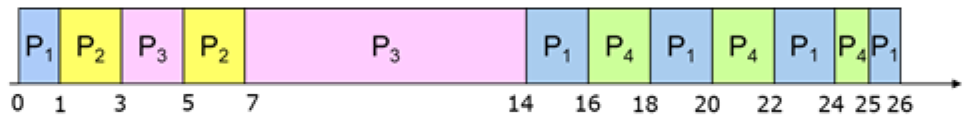
   The short-term scheduler selects the higher-priority process PH and allows it to use the CPU. PH tries to enter the critical section but must wait because PL is already in the critical section. Since PL has lower priority, it never gets a chance to use the CPU to finish its job in the critical section. As a result, PH is never able to enter the critical section.

3. (1 pt.)

   - CPU utilization
   - Throughput
   - Turnaround time
   - Average waiting time
   - Response time

4. (1.5 pt.)

(Answer)



Average waiting time = ((13+2+2+1)+2+(1+2)+(13+2+2)+/4 =

Average Turnaround time = (26 +(7 – 1) +(14 -2) + (25 -3))/4 =

5. (1 pt.)
   (Answer)
   - 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.
6. (0.5 pt.)
   (Answer)
   - When athread 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.

7. (2 pt.)
(Answer)

Let's assume the following initial conditions: `empty = 0`, `full = N`, and `mutex = 1`.

The producer is scheduled to run: it produces an item, performs a down operation on `mutex` (now `mutex = 0`), and then attempts to perform a down operation on `empty`. Since `empty = 0`, the producer is unable to complete the down operation and goes to sleep, waiting on the `empty` semaphore.

Next, the consumer is scheduled: it performs a down operation on `full` (now `full = N-1`), and then attempts to perform a down operation on `mutex`. Since `mutex` has already been locked by the producer, the consumer cannot complete the down operation and goes to sleep, waiting on the `mutex` semaphore.

Now, both the producer and consumer are stuck in an indefinite sleep state, causing a deadlock.

8. (2 pt.)
Answer)

Initially, `In = false`.

There are two processes, P_0 and P_1:

- P_0 is scheduled and attempts to enter the critical section.
- P_0 reads `In = false` but then times out. Its status changes from running to ready.
- P_1 is scheduled next and tries to enter the critical section.
- P_1 reads `In = false`, sets `In = true`, and enters the critical section.
- Some time later, while in the critical section, P_1 times out, changing its status from running to ready.
- P_0 is rescheduled and attempts to enter the critical section. Since P_0 had previously read `In = false`, it sets `In = true` again and enters the critical section.
- Now both P_0 and P_1 are in the critical section, violating the mutual exclusion condition.