Monina Atif Dar

1. T_1 : Inserts 'N'
2. T_1 : Inserts 'a'
3. Preemption – cuz not Thread safe
4. T_2 : Inserts 'G'
5. Preemption
6. T_1 : Inserts 'z'
7. T_1 : Inserts 'o'
8. Preemption
9. T_2 : Inserts 'u'
10. T_2 : Inserts 'l'
11. T_2 : Inserts 'a'

Q2 Best suited: Semaphores

Semaphores is best suited as it doesn't have spin lock problem so the CPU cycles are not wasted. It's implemented in hardware so it's fast and instructions are atomic which is another plus point.

Q3 My second choice would be Hardware Test and Set instructions because they are atomic. ~~so mutual exclusion will be taken care of.~~ And they are in hardware so they would be fast.

Q4 Because semaphores don't have the problem of 'busy-waiting'. The Thread that has to wait is sent to blocked state until it can be given turn. Whereas in Test and set the waiting thread does 'busy-waiting' so it wastes cycles of CPU.

Q5 I am assuming both Thread 1 and Thread 2 are running on same core so schedular has to give turns to both of them concurrently. For example, schedular gives turn to Thread 1 and then puts Thread 1 in waiting state and schedules Thread 2.
If the preemption does not occur or let's say Thread 1 is being executed on one core and Thread 2 is being executed on a separate core then my answers to the questions above will be incorrect.