4 problems. 80 points. Closed book. Closed notes. No electronic device. Write your name above.

1. [20 points] Jobs A, B, C have the following arrival times and service durations (in seconds):

- A: arrival time 0.0; service duration 3.5.
- B: arrival time 0.5; service duration 3.0.
- C: arrival time 3.5; service duration 2.5.

a. Assume fifo ready queue ("runnable queue" in GeekOS), no pre-emption, and zero context switch time. Complete the following table with a row for each successive service interval; each row indicates the interval and job being served.

Start time	Job served
0.0 – 3.5	A

b. Assume round robin with 1 second quantum, fifo ready queue, and zero context switch time. Complete the following table with a row for each service quantum. Indicate when a job departs.

Start time	Job served
0.0 – 1.0	A

2. [30 points] This question concerns GeekOS.

a. At the end of GeekOS initialization (and before the user does anything), how many threads exist and what is each thread doing.

b. During GeekOS initialization, `Init_Keyboard` installs an interrupt handler, but `Init_Screen` does not. Why not?

c. During GeekOS initialization, does `Init_IDE` have to install an interrupt handler? Explain briefly.

d. In GeekOS, from an interrupt occurrence to the interrupt handler being executed, the CPU does an action and then executes code involving `Handle_Interrupt`, `g_entryPointTable`, `s_IDT`, `g_interruptTable`. Write down the order in which these are done and briefly state happens in each.

3. [20 points] You are given buffer `buff` of max size `N` items and the following non-blocking functions: `num()`, returns the number of items in `buff`; `add(x)`, adds item `x` to `buff`; and `rmv()`, removes and returns an item from `buff`. Initially `buff` is empty.

Obtain functions `enQ(x)` and `deQ()` that satisfy the following requirements.

1. They can be called by multiple threads simultaneously.
2. Semaphores are their *only* synchronization construct (no atomic read-modify-write, no disabling interrupts, no access to PCBs, no wait/wakeup, etc.). No busy waiting.
3. `enQ(x)` calls `add(x)` exactly once, waiting if `num() = N` holds.
4. `deQ()` calls `rmv()` exactly once, waiting if `num() = 0` holds.
5. If a thread is in `enQ` and `num() < N` holds, then an `enQ` invocation returns.
6. If a thread is in `deQ` and `num() > 0` holds, then an `rmv` invocation returns.

Be neat and clear. You lose points if I can't understand your code in a reasonable time.

4. [10 points] *This extends problem 3.*

You are given `buff`, `num()`, `add(x)`, and `rmv()` as in problem 3.
Obtain functions `enQ(x)`, `deQ()` and `enQ2(x)` that satisfy the following requirements.

1 – 6. Same as in problem 3.
7. `enQ2(x)` calls `add(x)` exactly twice, waiting if `num()` $\geq$ `N-1` holds.
8. If (a thread is in `enQ2` or`enQ`) and (`num() < N-1` holds), then a thread returns from `enQ2` or `enQ`.

Be neat and clear. You lose points if I can't understand your code in a reasonable time.