EXERCISE #1

Suppose that a processor (single-core processor) is executing the following two instructions in
the two processes. If the initial value of variable ‘A’ is 100 at the beginning (before this
processor executes any of the two statements), what are the possible values in ‘A’ after this
processor executes the two statements (once for each)? Show all the possible values in ‘A’.
EXERCISE #2

An implementation of a circular FIFO queue for multiple producers and consumers are attached below. Regarding the implementation:

**Question #1**: Is it necessary to have “wait (mutex)” and “signal (mutex)” in the producer processes? If NO, explain why not. If YES, explain why we need them.

**Question #2**: Is there any merit in using two different “mutex semaphore” (one for the producers and the other for the consumers)? If yes, describe what is the merit? If not, why not?

**Question #3**: Is the implementation “starvation free” for both producers and consumers? If NO, explain why it is not. If YES, explain why we it is.

APPENDIX:

```c
#include N 100 // the queue size

shared memory int CQ[N]; // the circular FIFO queue
shared memory int TOP = 0; // pointer to the top of the queue
shared memory int TAIL = 0; // pointer to the tail of the queue

semaphore mutex = 1; // a mutex semaphore
semaphore empty = N; // a counting semaphore
semaphore full = 0; // a counting semaphore
```

```c
void producer (void) {
    int new_item; // place holder for a new item to insert

    while (TRUE) {
        new_item = produce_new_item(); // generate a new item
        wait (empty); // make sure the queue is NOT full
        wait (mutex); // I should be the only one
        insert(CQ, TAIL, new_item); // insert the new item to the queue
        TAIL = (TAIL + 1) % N; // update the Top pointer
        signal (mutex); // I am done!
        signal (full); // (full) = (full) + 1
    }
}
```

```c
void consumer (void) {
    int new_item; // place holder for a new item to insert

    while (TRUE) {
        new_item = remove (CQ, TOP); // remove one item from the queue
        TOP = (TOP + 1) % N; // update the Top pointer
        signal (mutex); // I am done!
        signal (empty); // (empty) = (empty) + 1
        use the new_item (new_item); // use the new item
    }
}
```
EXERCISE #3

For the following questions, answer by TRUE or FALSE. You do not have to attach any explanation to your solutions for these questions.

(1) If a processor has instructions to disable and enable any interrupts and if processes running in the USER MODE could disable and enable all interrupts using the instructions, such as hardware and software interrupts, we do not need mutex semaphores to avoid race conditions.

   [TRUE    FALSE]

(2) Since an operating system is a middleman between user processes and hardware resources (I/O devices), the operating system kernel should know how hardware devices execute the low-level commands to them.

   [TRUE    FALSE]

(3) For batch systems, FCFS and SJF scheduling algorithm can be used, while RR and SRTF can’t be used.

   [TRUE    FALSE]

(4) Printer spooling is a technique that improves process execution time in “batch system”.

   [TRUE    FALSE]

(5) Thrashing is caused by short-term process scheduling algorithms, mainly when they perform poor scheduling decisions (i.e., they did not select the best process to be executed next).

   [TRUE    FALSE]

(6) If process starvation happen, process deadlock will eventually happen.

   [TRUE    FALSE]

(7) Suppose a system where processes progress as they interact with many I/O devices. If we compare the batch systems and multitasking operating systems, batch systems will be faster in executing all the programs submitted (let us assume that many programs are submitted by a human user for execution) since batch systems will not perform as many context switching as multitasking operating systems do.

   [TRUE    FALSE]
(8) SJF process scheduling algorithm can be as good as SRTF in terms of throughput, but SJF can never be better than SRTF in terms of throughput.

[TRUE  FALSE]

(9) System calls are essential components in operating systems just because they provide “abstractions” to human users and user application processes.

[TRUE  FALSE]

(10) In the dining philosopher problem, if there are 100 philosophers on a table, the maximum number of philosophers who can eat at the same time is 50 philosophers.

[TRUE  FALSE]