(1) Which process scheduling algorithms can cause “process starvation” (select all that apply)?

(a) FCFS
(b) RR
(c) SJF
(d) SRTF

Note: Please clearly indicate which algorithms can cause “process starvation”: my suggestion would be to put a circle on the alphabet for the algorithm(s) you select.

Answer keys: (c) and (d)

(2) What are the two inefficiency problems in “processes”?

(i) High overhead in inter-process-communication (IPC)
(ii) High context-switching overhead in multitasking using processes

(3) How do threads reduce the high context switching overhead in processes?

Threads reduce the high context switching overhead (for thread to thread context-switching) by switching only a part of a PCB (the whole PCB will NOT be switched, but only the information that is unique to each thread). Thus, the time to switch a processor from a thread to another thread in the same process will be faster.
(4) Which of PCB (global PCB) or TCB (private PCB for each thread) do processor registers belong to (3 points)? Briefly (but with a proper emphasis on the essential reason) justify your opinion (7 points).

- PC (Program Counter) register should be in TCB (not PCB).
- PC register should be in each TCB, since the program codes (i.e., machine codes or processor instructions) each thread executes are different for each thread (or “each thread executes different processor instructions).

(5) As we discussed in the classroom, “threads” are introduced after many system programmers were using “processes” for multi-tasking (we even discussed that “threads” were introduced to avoid two problems in “processes”). After all, while “processes” and “threads” have many things in common (and “threads” seem to be better than “processes”). Then why do we still use “processes” (mention at least two different reasons)?

Any program that require a higher-level of “data privacy” and “robustness*” should be implemented using processes, instead of threads. The followings are some examples:

**Those that require a high level of “data privacy”:**

- On-line banking network servers
- E-commerce network servers
- Time-sharing servers (e.g., os.cs.siue.edu UNIX server)

**Those that require a high level of “robustness”:**

- On-line banking servers
- Account payroll software (and any accounting software)

*: the term, “robustness (for computer systems)”, means how a whole computer system is resistant to a crash of the whole system. For example, if a small or a minor issue (i.e., software bug) in one of the processes (or threads) can crash an entire computer system, we call such systems “weak robustness”.