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

- Processes are slow when they share data using IPC (shared memory) (since IPC requires an operating system to involve).
- Processes are slow in performing a context switching (since the whole PCBs need to be switched).

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

Threads reduce the overhead in performing context switching by not “switching” the entire PCBs. Threads perform context switching by switching only TCBs, which are smaller than PCBs. Switching smaller amount of data reduces the time needed for each context switching.

(3) Can threads in a process share global variables in other processes? If yes, briefly describe how. If not, briefly explain why not.

No. Global variables in a process is still protected from other processes.
(4) What information must be managed in TCB (thread control block, also known as “private PCB”) and what information should be still in the PCB (also known as “global PCB")?

The information managed by TCBs are those that are unique to each thread, while the information in PCBs are those that are common to all threads in a process.

(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)? If necessary, do your own research.

Processes are popularly used for application programs, where data protection or robustness of the programs is the highest priority.