(1) Why is it difficult to eliminate the condition of “hold & wait” (by applying “request all after you drop what all what you currently hold” method) to prevent a process deadlock from occurring?

If we apply “request all after you drop what all what you currently hold” method, the resources dropped by processes can be “stolen” by other processes that do not need a fewer resources. This will cause process starvations.

(2) Why is it difficult to eliminate the condition of “mutual exclusion” to prevent a process deadlock from occurring?

It is difficult since some resources, such as printers and DVD-burners, inherently require to be “mutually exclusive”.
(3) How do threads reduce the high context switching overhead in processes?

Context-switching threads (those in the same process) can be performed by switching TCB (Thread Control Block), instead of switching PCB (Process Control Block). Since TCBs are smaller than TCB’s, switching TCB’s will be faster than switching PCB’s. This reduces the high context switching overhead for threads.

(4) Show the typical internal structure of a process that consists of multiple threads.

(5) Which of the following items in the PCB for a process should belong to the global PCB or TCB (private PCB)?

(a) Processor registers (T)
(b) Program Counter (PC) register (T)
(c) User ID (P)
(d) Process ID (P)
(e) The list of opened files (P)
(f) The list of the assigned I/O devices (P)
(g) Stack Pointer (SP) register (T)