(1) What are the advantages in the kernel-mode threads?
   
   - Preemptive thread scheduling is possible (fairness).
   - A bug in a thread will not crash all threads in the same process (Robustness).

(2) What are the advantages in the user-mode threads?

   - Source code files developed for a system can be used by another with little or no modification) (portability)
   - no processor-mode change when a context-switching happens from a thread to another thread in the same process (speed = faster thread executions)

(3) What are the four necessary conditions for a process deadlock to occur?

   - Preemptive resources
   - Not mutually exclusive resources
   - Hold & wait
   - Circular wait
(4) 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 one reason)?

There are some applications that need security. Processes are better than threads for making sure that data in a process is protected from other threads.

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

Some resources, such as printers and DVD writers, inherently require mutual exclusion.