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

- The primary advantage in the user-mode threads is portability. Since the OS kernel does not have anything to do with threads, as long as a standard thread library is available to different operating systems, you most likely can use your source code files (for threads) in many different operating systems.

- The ability for customizing the mechanisms for managing threads. Since managing threads are performed as a part of your process, over-riding those mechanism is relatively easier (than the kernel-mode threads).

(2) Complete the following table that compares the user-mode and kernel-mode threads.

<table>
<thead>
<tr>
<th>Factors</th>
<th>User-Mode</th>
<th>Kernel-Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preemptive thread scheduling</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>Robustness</td>
<td>POOR</td>
<td>GOOD</td>
</tr>
<tr>
<td>Execution speed</td>
<td>FAST</td>
<td>SLOW</td>
</tr>
<tr>
<td>Portability</td>
<td>GOOD</td>
<td>POOR</td>
</tr>
</tbody>
</table>

(3) How is “process deadlock” different from “(process) starvation”?

In process starvation, the issue is “fairness”, while that for “process deadlock” is robustness.

**In process starvation:**
- There is always at least one process (or thread) that is making progress.
- If a process(es) that is monopolizing resource leaves, there is a chance for the starved process to make progress.

**In process deadlock:**
- Everyone (all processes that are involved in a process deadlock) stops running.
- There is no way hope for any process (those that are involved in a process deadlock) to make progress (once it happens, it never disappears until some action is taken).
(4) What are the four necessary conditions for a process deadlock to occur?

1. Mutual exclusion (mutually-exclusive resource)
2. Hold & wait (a process can hold resource while it waits for any unavailable resources)
3. Circular wait
4. Non-preemptive resources

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

Eliminating “mutual exclusion” to prevent process deadlock from happening is sometimes impossible, since some I/O devices are “mutually-exclusive I/O devices by their nature (e.g., printers, DVD-burners, and etc.)