blocked queue while the process interrupted by the timer will be put onto the ready queue. The system call is a synchronous event (software interrupt) while the timer interrupt is an asynchronous event (hardware interrupt).

8) Describe the problems associated with threads with respect to OS functions and implementation. What are the tradeoffs in managing threads in user space verses kernel space ?

One problem deals with the fork() system call. If a multiple thread process fork()'s and the OS creates copies of the threads in the child, then problems are created, for example, if one of the threads is blocked for keyboard input. Which thread does the OS deliver the keyboard input to ? Another problem is dealing with updates to data structures shared among threads at the OS level, such as the data structures associated with memory allocation and open files. Error reporting, signal handling and stack management also introduce problems in the implementation of OS functions.

The advantage of managing threads at the user level is that switching between threads is much faster than it is if a system call is necessary. The disadvantage is that when one thread blocks for I/O, the kernel blocks the entire process since it is not aware of the other threads.

9) Describe the conditions for race conditions and describe how a shared memory variable value may become corrupt under such conditions.

A race condition occurs when two or more processes are reading and writing shared memory and the final result is dependent on the interleaving sequence of the processes.

Process A moves the value of shared memory variable 'shared' into a register (maybe because it is going to test some condition on its value and then update it based on the result of the evaluation). The timer fires and process B moves and updates the value of 'shared'. When process A is rescheduled, it operates on old value of shared and overwrites the updated value written by process B.

10) Describe why it is not possible to change the while statement in Peterson's solution to

```
while (turn != process && interested[other] == TRUE)
```

Process A executes 'interested[process] = TRUE', 'turn = process' and falls through the while loop shown above.

While in the critical section, the timer fires and process B executes 'interested[process] = TRUE', 'turn = process' and falls through the while loop shown above.

Both process are in the critical section.

A more interesting question here (the one I wanted to ask) is under what sequence of events does the above solution succeed in providing mutual exclusion ?

1) Define multiprogramming and explain how it is able to increase CPU utilization.
Multiprogramming is an activity in which several memory resident processes are run concurrently (interleaved in time) by the OS. Multiprogramming increases CPU utilization because the OS can switch to another ready process whenever the currently running process blocks for I/O.

2) Define SPOOLing and explain how the OS uses SPOOLing and multiprogramming to increase both CPU utilization and I/O device utilization.
SPOOLing is defined as Simultaneous Peripheral Operation On Line. In SPOOLing, the OS buffers input and output to disk in order to overlap the I/O of one process with its computation. Multiprogramming additionally allows the computation of other jobs (and the use of other I/O devices) to be overlapped.

3) Assume, in a multiprogrammed batched system, that the OS only switches to another process when the running process blocks for I/O. What additional state transition is necessary in a time sharing system and why is it necessary ?
The additional state transition is *Running to Ready*. In a time shared system, multiple users are sharing the same hardware. This additional transition is necessary in order to improve system response time and provide fairness in the use of the CPU to the users.

4) Why are system calls necessary and describe the general categories into which all system calls can be placed.
System calls are necessary in order to allow applications to access OS resources (e.g. hardware). This allows the OS to enforce protection and fairness in the use of these resources among the processes competing for them. The general categories include process management, signals, file management, directory and file system management, protection and time management.

5) Describe the capabilities of the virtual machine provided by the IBM VM/370 OS and the virtual 8086 machine provided by the Windows OS on Intel machines.
The VM/370 runs on the bare hardware and does the multiprogramming, providing not one, but several virtual machines to the next layer up. The virtual machines are exact copies of the bare hardware. The Windows OS, which must be run on an 80386 and higher, provides an exact copy of a different machine, the 8086.

6) Why is it necessary to provide a hardware mechanism for disabling interrupts on the 8259A and the CPU separately ? (Think about this in terms of ISRs).
It is necessary in order to allow ISRs that process interrupts of lower priority to be handled while an ISR, which processes an interrupt of higher priority, is running. This allows the OS to establish the priority of the interrupts directly within each of the ISRs.

7) Describe the differences between a system call for input that is not available and the timer ISR handling by the OS for a process whose time slice has expired.
The system call for input that is not available will cause the process to put onto the