Q1: The following actions, do they require the OS to use kernel mode or user mode is sufficient? Explain.

- Starting a new process.
- Multiplying two floating numbers stored in an application's memory.
- Writing the results of the multiplication to a temporary log file on disk.

Starting a new process

The OS needs to switch to kernel mode in order to start a new process. Always when a new process is created, the OS must allocate different resources like memory and CPU time. This can only be done in kernel mode. Also, the OS sometimes needs to make changes to data structures and databases that cannot be accesses in user mode. Moreover, launching a new process requires running the fork() system call, which moves the CPU from user mode to kernel mode. Therefore, starting a new process requires the OS to use kernel mode.

Multiplying two floating numbers stored in an application's memory

Most of the time, multiplying two floating-point numbers stored in an application's memory does not require kernel mode. A simple example is if we make a program to multiply two floating-point numbers, the operation can be completed entirely within user mode. However, if the application tries to access memory that it is not permitted to access , the OS will transition to kernel mode.

Writing the results of the multiplication to a temporary log file on disk

The OS needs to be in kernel mode when writing the results of a multiplication operation to a temporary log file on disk. The application must first ask the OS through a system call for permission to access the file system before it can write data. The OS then allocates resources like disk space which requires kernel mode. Moreover, writing to disk requires shifting data from primary storage to physical memory, which can only be done in kernel mode.

Q2: Explain the purpose of a system call. Describe how control passes to the kernel when executing a system call.

A system call is a request made by a program running in user mode to the OS kernel running in privileged mode. When a program needs to perform an operation that requires access to system resources such as file I/O, network communication, or memory management, it must make a system call to the operating system to perform that operation. **The purpose of a system call** is to provide a safe and controlled interface for user-level programs to interact with the OS.
The program executing in user mode makes a system call by invoking a system call instruction. The system call then generates a software interrupt, known as a trap instruction. **The trap function** causes the processor to switch from user mode to kernel mode and execute an

interrupt service routine (ISR) in the OS kernel. This allows the kernel to perform privileged operations. The kernel interrupt handler then saves the user mode state and the program counter and registers, onto the stack. The kernel performs the requested operation, using the privileged access it has. Once the operation is completed, the kernel stores the result in a location such as a register. In the end, the return-from-trap instruction is executed. The job of this function is to transfer control back from kernel mode to user mode after the completion of the system call.