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

● Starting a new process.

This action requires kernel mode since when creating a new process and starting it, you need to ask the kernel for certain functionalities (fork(), exec()...)

● Multiplying two floating numbers stored in an application's memory.

Happening in user mode - for example compiler is an application in which this action can be performed, and as we said, applications are running in user mode

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

Even though this task is similar to the previous one, this action is happening in kernel mode since writing to a temporary log file on disk is an action that is restricted for the users.

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 method that allows a process to communicate with the operating system. It is a programmatic method in which a computer program requests a service from the kernel of the OS. A typical OS, in fact, exports a few hundred system calls that are available to applications. Because the OS provides these calls to run programs, access memory and devices, and other related actions, we also sometimes say that the OS provides a standard library to applications.

When a user program invokes a system call, a system call instruction is executed, which causes the processor to begin executing the system call handler in the kernel protection domain. Calls the function that implements the system call that was requested. The system loader keeps track of which functions are used for each system call in a table. Because system calls are run in the kernel protection domain, they have more privilege than the calling thread. After the function implementing the system call has performed the requested action, control returns to the system call handler.