Disclaimer: these questions are not in the style of exam questions. They are meant to make you think about possible trade-offs and different possible solutions to problems that we face when building a storage system, and to make you think about whether you got the "big picture".

Quiz 1 - I/O

USB mouse	Different designs are conceivable; for modern USB mice, the USB controller keeps polling the mouse, and if a button click or movement is detected by the USB controller, then the controller issues an interrupt to the CPU. This way, the CPU isn't busy dealing with the mouse unless anything actually happens.
DMA	The DMA processes commands that are issued by the CPU, and those commands specify which memory addresses have to be written to/read from which addresses in storage.
System crashes	Most system crashes are the result of poorly written device drivers. The producers of operating systems that don't control which hardware users will attach offer an API for device drivers to use, and developers who write those drivers can introduce bugs by using the API incorrectly, or by making the drivers do things within a device that cause errors. Since driver code is executed in kernel mode, drivers can easily cause problems for the system that way.

Quiz 2 - Hard Disks

Crash while writing	Modern hard drives are built in a way that even if power fails while a sector is being written, the write completes.
OS and physical location	An HDD does not expose its geometry to the OS, so the OS does not know where blocks physically are.
Storage capacity of tracks	For geometric reasons, the outer tracks are longer than the inner tracks, and since we can utilize that extra length with additional blocks, different tracks do not have the same storage capacity.
Disk scheduling	Both the OS and the disk itself can schedule requests. The OS can decide which of the requests that it wants to issue it will issue first, and the disk can rearrange the requests it receives as well.

Quiz 3 - Files and Directories

Windows shortcut	A shortcut on Windows is a small file with the file extension `.lnk`; that file extension is never shown in Explorer, but can be seen in a command line using the `dir` command.
What happens when mounting	The original /dir becomes invisible, until sda2 is unmounted again.
Mounting to C:/dir	To our surprise, this is possible.

File descriptors	Every process has the three file descriptors 0, 1, 2 (standard input, standard output, standard error) by default. File descriptors that get opened are assigned consecutively, so if two different processes each open just one file, each process will have the number 3 as the file descriptor for that file. Those two file descriptors are not the same object, but contain the same information.
lseek/fseek	Those are system calls to move the logical position in a file. `fseek` is the name in C/C++, `lseek` is the POSIX name.
Same file in two directories	If a file is represented by an inode, and by “having the same file twice” we mean that we have two hard links, then the data of the file is stored only once. On the other hand, if we copy a file, then we will need two physical copies of the same data in case one of the two files gets edited so the other one remains unaffected. We might consider doing this lazily: we could only actually copy the data once an edit is supposed to be written to disk, but this would delay the cost of copying to the moment of editing.
Hard link to directory	Most operating systems do not allow hard links to directories as this might introduce cycles in the directory tree. Many tools rely on the directory tree not containing any cycles, thus the easiest way to conform to that is not allowing hard links to directories. Soft links to directories are allowed on the other hand, and the reason is that they can easily be ignored in case cycles are detected/might be an issue as they can be distinguished from hard links easily. Distinguishing between two hard links with regard to which one is “the real one” is not possible, since both are equally “real”.
	Programs could be adapted to detect hard link cycles in the directory tree as well. Some operating systems incorporate file systems that can deal with that, or allow hard links to directories, but only ones that are in different subtrees than the one the hard link is in. The easier design choice made in most operating systems however was to not allow hard links to directories.

Quiz 4 - File System Implementation

Finding free blocks	Some file systems use a free list, some use bitmaps, others might use different ways.
Physical location of file	The OS can choose which logical blocks it wants to write to, and the disk can choose which physical block to write a logical block to. For HDDs, the logical-to-physical translation is not strictly necessary; however, modern HDDs can mark faulty blocks as dead, and a write to a dead block has to be relocated to a live block, which a logical-to-physical translation can do.
Number of inodes per file	Each file is represented by exactly one inode. A hard link to a file is an entry in a data block of a directory that associates a name with an inode number, so multiple hard links will all point to the same inode.

inode size	Inodes have a fixed size. Inodes for larger files will make use of indirect pointers, which point to data blocks that contain pointers of one level lower. For example, a doubly-indirect pointer points to a data block that only contains singly-indirect pointers. Larger files will take up more space in the data region due to their larger size and the overhead of data blocks used for indirect or direct pointers, but no more than one inode is involved.
inode and file name	An inode does not contain a file name. The only places where names to a file exist are the data blocks of directories; those contain pairs of "file name:inode number" that are used to resolve paths. Different names might point to the same inode.
Path length and disk accesses	Resolving `/dir/dir/dir/dir/foo.txt` costs more disk accesses than resolving `/foo.txt` since every subdirectory has to have its data block(s) read, then the inode of the next subdirectory has to be read to find its data block(s),... all the way down to the lowest subdirectory. However, modern operating systems cache recently resolved paths in main memory (i.e. RAM), and so only the first resolution of either path would cost any disk accesses at all, assuming neither path leaves the cache.
Block size in FAT	The blocks in a single partition are the same size.

Quiz 5 - SSDs

Flash faster than disks	On average, flash disks are significantly faster than disks, in particular for random access. It is possible that a very high end HDD might be on par with or slightly faster than a very old and slow SSD in some situations, but comparing similar market segments, SSDs win.
SSD wearout from reads	Reading a page of an SSD does not cause any wear. However: many operating systems attach metadata to files that stores information about when a file was last accessed, and if this is the case, then a read access - while not causing any wear itself - will trigger a write access to update the most-recently-accessed value, and that write will cause wear. For this reason, most-recently-accessed information will sometimes be disabled by administrators when using SSDs.
Changing a file	When changing a file, even if only a small counter, a write access issued to the SSD will typically not overwrite the page in which that portion of the file previously resided (see FTL).
More than 1 TB?	SSDs contain more actually usable space than is exposed to the user. This extra space is used for wear leveling.
Log-structured	The name comes from the practice of logging write requests by appending them to the end of a log.