Question 4.1:

file descriptor: In Unix and related computer operating systems, a **file descriptor** (**FD**, less frequently **fildes**) is an abstract indicator (handle) used to access a file or other input/output resource, such as a pipe or network socket. File descriptors form part of the POSIX application programming interface. A file descriptor is a non-negative integer, generally represented in the C programming language as the type int (negative values being reserved to indicate "no value" or an error condition).

Inode: The **inode** (index node) is a data structure in a Unix-style file system that describes a file-system object such as a file or a directory. Each inode stores the attributes and disk block locations of the object's data.[1] File-system object attributes may include metadata (times of last change,[2] access, modification), as well as owner and permission data.

Directories: In the FHS, all files and directories appear under the root directory /, even if they are stored on different physical or virtual devices. Some of these directories only exist on a particular system if certain subsystems, such as the X Window System, are installed.

Soft and hard links: A link in UNIX is a pointer to a file. Like pointers in any programming languages, links in UNIX are pointers pointing to a file or a directory. Creating links is a kind of shortcuts to access a file. Links allow more than one file name to refer to the same file, elsewhere.

There are two types of links :

1. Soft Link or Symbolic links
2. Hard Links

These links behave differently when the source of the link (what is being linked to) is moved or removed. Symbolic links are not updated (they merely contain a string which is the pathname of its target); hard links always refer to the source, even if moved or removed.

For example, if we have a file a.txt. If we create a hard link to the file and then delete the file, we can still access the file using hard link. But if we create a soft link of the file and then delete the file, we can't access the file through soft link and soft link becomes dangling. Basically hard link increases reference count of a location while soft links work as a shortcut (like in Windows)

1. Hard Links

- Each hard linked file is assigned the same Inode value as the original, therefore they reference the same physical file location. Hard links more flexible and remain linked even if the original or linked files are moved throughout the file system, although hard links are unable to cross different file systems.
- ls -l command shows all the links with the link column shows number of links.
- Links have actual file contents
- Removing any link, just reduces the link count, but doesn't affect other links.
- We cannot create a hard link for a directory to avoid recursive loops.
- If original file is removed then the link will still show the content of the file.
- Command to create a hard link is: $ ln [original filename] [link name]

2. Soft Links

- A soft link is similar to the file shortcut feature which is used in Windows Operating systems. Each soft linked file contains a separate Inode value that points to the original file. As similar to hard links, any changes to the data in either file is reflected in the other. Soft links can be linked across different file systems, although if the original file is deleted or moved, the soft linked file will not work correctly (called hanging link).
- ls -l command shows all links with first column value l? and the link points to original file.
- Soft Link contains the path for original file and not the contents.
- Removing soft link doesn't affect anything but removing original file, the link becomes "dangling" link which points to nonexistent file.
- A soft link can link to a directory.
- Link across filesystems: If you want to link files across the filesystems, you can only use symlinks/soft links.
- Command to create a Soft link is: $ ln -s [original filename] [link name]