1. Please explain the use cases of soft and hard links. When should we use a soft link, and when is a hard link more appropriate?

- **Soft link (symbolic link)** points to a file path.
- Use when: linking across different filesystems, or when you want the link to break if the original is deleted.
- **Hard link** points to the same inode.
- Use when: you want multiple filenames referring to the same data.

1. Describre the inode relationship of soft links. How do inodes work with soft and hard links? Please show inode numbers and explain how the inode accesses the data on the hard disk.

- **Soft link**: has its own inode which stores a path to the target.

```
ls -li file link
12345 -rw-r--r-- 1 user user 0 file
12346 lrwxrwxrwx 1 user user 4 link -> file
```

- **Hard link**: shares the inode with the target file.
- Both point to the same data blocks.

3. What is the `/opt` directory used for? How does it differ from `/etc` and `/sbin` in terms of purpose and usage?

- **/opt**: Optional software packages (e.g., custom or third-party apps).
- **/etc**: System configuration files.
- **/sbin**: System binaries for administration (usually root-only tools).

3. Compare the locate and find commands. Which one is faster, and why?

   The `locate` command searches a prebuilt database, making it faster, while `find` scans the filesystem in real-time, offering more flexibility. `locate` is quicker due to database lookup but may be outdated.
4. Please explain how your device's `updated.conf` is configured. How does it affect the behavior of the `locate` command?

- `updatedb.conf` (usually in `/etc/updatedb.conf`) defines what directories to include/exclude from indexing.
- It affects what `locate` can find. For example, `/tmp` or `/mnt` might be excluded.

5. What is the difference between bashrc and bash_profile? What is the purpose of /etc/bash_completion?

- **.bashrc**: Runs for **non-login** interactive shells.
- **.bash_profile**: Runs for **login** shells.
- **/etc/bash_completion**: Provides autocomplete rules for many CLI tools (e.g., `git`, `docker`).

7. What kind of information is stored in bash_logout and bash_login? What are their typical use cases?

   The `~/.bash_logout` runs commands on shell exit (e.g., clear history), and `~/.bash_login` sets environment for login shells (e.g., `PATH`).
8. Please demonstrate how to customize the command prompt (PS1) in the bashre file.

```
# In ~/.bashrc
PS1='\u@\h:\w\$ '
```

This sets the prompt to: `username@hostname:current-directory$`

9. Write a Bash script that takes a webpage URL as input and downloads all the URLs (links) found on that page.
   It is not wise to scrap HTML with regex, and it is better to use a programming language that has a library for that.
   first we install `lynx` tool, then we use it in our script:

```
read url_input;
lynx -dump -listonly "$url_input" | grep https | awk '{$1="";
sub(/^\.\s*/, ""); print}' | wget -r -i -
```

```
read url_input;
lynx -dump -listonly "$url_input" | grep https | awk '{$1="";
sub(/^\.\s*/, ""); print}' > list_of_urls.txt

while read url; do read filename; wget -O $filename $url; done <
list_of_urls.txt
```

10. Please explain the use of the exit command and other common exit-related statements in Bash scripting.

- `exit`: Ends a script or shell session.
- `exit 0`: Success.

- `exit 1` (or any non-zero): Failure.
- `return`: Used inside functions to return a value to the caller.

  There are some exit codes with special meaning that can be find here.
  https://tldp.org/LDP/abs/html/exitcodes.html#EXITCODESREF