Question 1:

Expanding on the implementation of the `ls` command and the `redirection operator` shown in class, using system calls, you need to implement the long form of the `ls` command (`ls -l`) on a specified directory and redirect the output to a text file (`list.txt`).

Explanation:

For example, your **.c** executable name is `myls` and you give a command line argument as the directory you want to perform the `ls` on. Your prompt will look like this.

```
./myls /DSTN/TestFolder
```

This translates to `ls -l /DSTN/TestFolder > list.txt`

If no command line argument is given (`./myls`), do the ls command on the current directory.

```
ls -l . > list.txt
```

Hint:

- Important system calls: `opendir(), readdir(), stat(), dup2(), exec family` system calls. You should use more system calls as required.
- Implement the ls command first. Then use that executable as the parameter for the exec call to implement the redirection operation.

Question 2:

- Write a bash script or a python script to emulate a file system behavior. For example, a user can create a directory, create multiple files, write into those files, create subdirectories, create more files, read those files, create hard links for the files, delete the files, and finally delete the folder.
- Now use `strace` tool to trace the system calls issued while running the script and log those in an output file. You should only trace the system calls which are taught in class. (Hint: use the `-e` and `-o` flags).
- Write a **monitor (**in any language**)** that parses those system calls from the output file and tells the user in plain English language what is being done on the file system.
  For example:
  - `Created directory <directory name>`
  - `Created file <filename>`
  - `Read from file <filename>`
  - `Written into file <filename>`
  - `Renamed <filename> to <filename2>`
  - `Deleted the file`
  - `Created a hard link <link name> for <filename>`
  - `Deleted directory <directory name>`