Question 1 (1 point)

Random Question 1: Chmod Command [10 points]

Change the provided codebase to **support read and write permissions**. The syntax of your chmod command is:

```
chmod_EXAM <file_ID_in_active_file_table> <read_permission> <write_permission>
```

- The `file_ID_in_active_file_table` parameter is an integer that corresponds to the array cell index in the active file table where the user has previously opened the file.
- The `read_permission` parameter has two possible values: `r+` and `r–`. The r+ means the file can be read. The r- means the file cannot be read. If another parameter value is given, an error message will be displayed. `“ERROR: Bad syntax.”`
- The `write_permission` parameter has two possible values: `w+` and `w–`. The w+ means the file can be written. The w- means the file cannot be written. If another parameter value is given, an error message will be displayed. `“ERROR: Bad syntax.”`

Code hooks in the codebase

In `interpreter.c` add a function called `chmodCommandEXAM()` and make the necessary changes to connect the function to the `chmod_EXAM` command, similarly to what was provided to you for the seek, open and close functions. You have also been provided with two empty files, `DISK_driver_problem4.c` (and .h). Use these three places to implement the `chmod_EXAM` functionality. Feel free to expose other functions and data structures already existent in the codebase to your `DISK_driver_problem4` files. You should be able to complete your implementation by only modifying these three locations and the FAT table in `DISK_driver.h`. However, if you absolutely need to make modifications elsewhere in the codebase, make a note in the README file, explaining why.

Implementation details

- Add 2 Booleans to the FAT table in RAM and in the partition FAT table to keep track of read permission and write permission. The permissions are persistent.
- By default, when a file is opened for the first time, it has no read and no write permissions.
- The `chmod_EXAM` command will only change the permissions of the file is currently opened.
- Before a script's `read_EXAM` / `write_EXAM` command is executed a permission check must be carried out.
- The permission must be true for the operation to happen. If the permission is false and the test file tries to issue a write/read command, an error message is displayed: `“ERROR: Write permission denied.”` or `“ERROR: Read permission denied.”` The write/read command is then skipped, and the script continues executing.
- The `seek_EXAM` command does not check the permissions.

Testing the code

Test your code with the following test file:

```
1.  mount partition4 10 5
2.  open_EXAM 0 file1
3.  chmod_EXAM 0 r-
4.  chmod_EXAM 0 r- w+
5.  write_EXAM 0 [hello world test]
6.  close_EXAM 0
7.  open_EXAM 0 file1
8.  read_EXAM 0 var
9.  print_EXAM var
10. chmod_EXAM 0 r+ w+
11. read_EXAM 0 var
12. print_EXAM var
13. quit
```

The command on line 3 should display: `ERROR: Bad syntax.`
The command on line 8 should display: `ERROR: Read permission denied.`
The command on line 9 should display: `Variable does not exist`
The command on line 12 should display: `hello world test`

Notes:

- We will not test other corner cases. We will test your solution with the above test file and by looking at your source code to see if it was constructed correctly and whether it respects the expected programming style for this class.
- Your code will also be analyzed by MOSS for cheating.
- Your solution must run on mimi.

Marking breakdown:

- **2 points.** `read_EXAM` and `write_EXAM` commands validate permissions.
- **2 points.** Permissions are correctly stored in RAM FAT table.
- **2 points.** Permissions are correctly stored persistently in partition FAT table.
- **2 points.** Permissions loaded correctly when `open_EXAM` command used.
- **2 points.** `TESTFILE` runs to specification
- Your TA can remove **up to 3 points** from the total if the coding style is not respected.