Write a bash shell script, called **perm**, that satisfies the following requirements:

1. Usage: perm [ dir-name ]
2. **perm** displays the permissions of a directory and all the directories in the directory's absolute path. **perm** will also show other information about the directories, as the cursor is moved up and down the list of directory names. **perm** will also facilitate changing permissions of the directories, assuming that you are the owner of the appropriate directory or the superuser.
3. **perm** will accept one argument, the name of a directory, including the path if required.
4. If there is no argument specified, then the current directory will be the default.
5. If there is more than one argument specified, there will be an error message and the script terminated (exit status 1) with no full-screen display.
6. If the specified filename is not a valid, existing directory, there will be an error message and the script terminated (exit status 1) with no full-screen display.
7. **Please see the examples for the exact display format, and match this format as closely as possible**. Note that the screen is cleared before full-screen output is displayed. There are 3 spaces between the permission groups, and between the permission groups and the filename. There is one space between the permissions within a permission group.
8. Ignore the situation of a very deep directory structure, which would require a multi-screen display.
9. Do not let output lines wrap to the next line, ensure that all lines are truncated to the width of the screen size.
10. Once your program is running, you do NOT have to take into account changes in screen size. You also do NOT have to take into account changes performed from another process, such as changes in permissions of the displayed directories or changes in the displayed directory structure.
11. On initial display, the cursor will be on the first character of the specified (or default) filename. Information for this directory will also be displayed.
12. There are three lines showing the valid keys accepted during execution of the **perm** command, on the fourth, third, and second lines from the bottom of the terminal window, regardless of the terminal window size used.
13. After the initial display, the valid keys available are the following. Note that there should be no "enter" key required after any of them, and that they are lower case:
    * "k" will move the cursor to the line containing the filename above the current line. Also, information for the new directory level will be displayed, instead of the previous directory level. This key will have no affect if the cursor is already on the top-most filename, which will be the root.
    * "j" will move the cursor to the line containing the filename below the current line. Also, information for the new directory level will be displayed, instead of the previous directory level. This key will have no affect if the cursor is already on the bottom-most filename, which will be the specified (or default) directory.
    * "h" will move the cursor to the field immediately to the left of the current field. This key will have no affect if the cursor is already on the "r" permission position for "owner". The fields are the nine permission fields, and the filename.
    * "l" will move the cursor to the field immediately to the right of the current field. This key will have no effect if the cursor is already on the right-most field, which will be the filename.
    * "r" will set read permission to the current filename, for owner, group, or other, depending on which field the cursor is on. If the cursor is on any field other than one of the three "r" fields, then the "r" key will be ignored.
    * "w" will set write permission to the current filename, for owner, group, or other, depending on which field the cursor is on. If the cursor is on any field other than one of the three "w" fields, then the "w" key will be ignored.
    * "x" will set execute permission to the current filename, for owner, group, or other, depending on which field the cursor is on. If the cursor is on any field other than one of the three "x" fields, then the "x" key will be ignored.
    * "-" will remove a permission to the current filename, for owner, group, or other, depending on which field the cursor is on. If the cursor is on any field other than one of the nine permission fields, then the "-" key will be ignored.
    * "q" will terminate the **perm** script. Note that the screen is not cleared, and the command prompt will appear on the bottom line of the terminal window.
    * <Ctrl>-c will have exactly the same effect as "q"
14. Any keys entered, other than the valid keys above, will be ignored.
15. If the user doesn't have authority to change a permission, then the "r", "w", "x", and "-" keystrokes will be ignored. If the permission could be changed, then the output display will reflect the new permission.
16. When moving down or up, the cursor will be on the same field on the new line as on the current one.
17. "Flashing" of the screen, during cursor movement, is OK. For example, when the cursor is moved up or down, you may redraw the entire screen. If you wish to eliminate the flashing, making the program more pleasant to use, you may do so, but it will slightly increase the complexity of your program.
18. If **perm** produced full-screen output, it must end with an exit status of zero. If there was an error, then there is no full-screen output, and the exit status should be one.
19. Ensure that your script is self-contained in one script file, called **perm**
20. Don't do any argument verification or any processing in **perm** other than that specified above. If you're interested in adding more capabilities, please do so in a different script.