Script Writing - this script is marked out of 100 marks total
This script is worth 17% of the total 25% for this exam

Design and then write (in that order!) the following **`bash`** script that will extract some data from a Unix-style password file and then use the data.

For best marks, your finished script on paper must have the appearance that you **designed** it before you started coding it. Many **insertions** and/or excessive **crossed-out** material will cost you marks. Write your PDL before you code!

Each of the steps below has been written so that you can code it **independently** of previous steps, using the given information. Do as much as you can.

Lines in the Unix password file have this 7-field colon-separated record format:

`userid:password:uid:gid:lastname firstname:home:shell`

Follow these steps to write this script:

1. *[Marks: 20]* **Document your code.** The script you write does not need an assignment label or purpose section; but, it must follow the other course script writing guidelines, especially with regard to the interpreter line, path, file creation mask, and **block comments**. Include the associated **step number** in each of your block comments.

   For best marks, document with block comments *all* the steps, even steps that you do not know how to code. (Leave the code empty.)

   Remember to **validate all inputs**. Issue appropriate useful, helpful **error messages** if you find incorrect, missing, or unusable input.

2. *[Marks: 12]* Script syntax: **`$0 fullname [ pathname ]`**
   Input: The script will take either one or two command line arguments.

   The first argument is a person's full name (e.g. **`"Ian D. Allen"`**), and it is mandatory. If the argument is missing, generate a useful, helpful error message and exit the script with status **1.** Transfer the first argument to a variable named **fullname**.

   The second command line argument is a password file **pathname**, and it is optional. If the second argument is present, transfer it to a variable named **pathname**. If it is missing from the command line, issue a good prompt and read the **pathname** from the user.

3. *[Marks: 3]* If the user's input is blank (the **pathname** is an empty string), set the **pathname** to be the usual Unix password file named **`/etc/passwd`**.
4. *[Marks: 5]* If the **pathname** is not readable, issue a useful, helpful error message and exit the script with status **2.**
5. *[Marks: 5]* If the **pathname** is not a plain file, issue a useful, helpful error message and exit the script with status **3.**
6. *[Marks: 5]* If the **pathname** file is zero size (is empty, contains no data), issue a useful, helpful error message and exit the script with status **4.**

7. *[Marks: 3]* Convert the text string in **fullname** to all lower-case letters.
8. *[Marks: 3]* Split the text string in **fullname** on white-space, and put the first field into a variable named **firstname**.
9. *[Marks: 3]* Split text string in **fullname** again, and put the *last* field (not the second field) into a variable named **lastname**.
10. *[Marks: 3]* Remove all the single-quote characters from the text in **lastname** (e.g. change **`o'donnell`** into **`odonnell`**).
11. *[Marks: 8]* Refer to the colon-separated, 7-field record format of the Unix pasword file, above. Process the password file named by **pathname** and match the user-entered **lastname** and **firstname** together against the field containing the same information in the password file. (Get the order of the two parts of the name right when you do the match.) The match must be an exact match; you may assume that the password file has exactly one blank between the last and first names in the name field. If the two user-entered names exactly match the corresponding two names in the name field in the password file, extract just the **`userid`** field from the matching line and save the userid into a variable named **userid**. Remember: The field separator in the Unix password file is the colon character ("**`:`**").
12. *[Marks: 6]* If the above search failed to match any name, issue a good, useful, helpful, fully-detailed error message and exit the script with status **5.**
13. *[Marks: 2]* Create a variable named **foo** that contains the pathname **/home/***uuu***/*foo*** where the string *uuu* is replaced by the **userid** just found in the password file.
14. *[Marks: 2]* Create a variable named **bar** that contains the pathname **/home/***uuu***/*bar*** where the string *uuu* is replaced by the **userid** just found in the password file.
15. *[Marks: 8]* If either the pathname in variable **foo** or the pathname in variable **bar** are not readable, plain files, issue a good, useful, helpful, fully detailed error message and exit the script with status **6.**
16. *[Marks: 6]* Count the number of lines of differences between the two files whose names are in **foo** and **bar** and display the count of the lines as follows:

    File "*fff***" and "***bbb***" difference = '***nnn***' lines.**

    where the two strings *fff* and *bbb* are replaced by the two file names, and the string *nnn* is replaced by the line count of the differences between the files.
17. *[Marks: 2]* If the count of the number of lines of differences between above two files is less than 10 lines, also output the line:

    `Almost same: < 10 lines of difference output.`
18. *[Marks: 4]* Display the first 15 lines of the file whose name is in the variable **foo**, with each line preceded by its line number.

Reminder: Did you follow all the CST8129 script coding conventions?