**Scripting Languages**

**Project**

Write a bash file that will output the answer each of the following questions

# Exercise 1

1. For this problem you will write a Shell script called todo.sh that allows a user to manage a todo list at the command line.

(Hint: think of applying the switch case of each option)

The todo.sh script can take numerous forms of command line arguments…

* ./todo.sh ***help***
* ./todo.sh ***add "task in quotes" [optional date string]***
* ./todo.sh ***del <number>***
* ./todo.sh ***list [optional date string]***

Any command line arguments that do not adhere to one of the aforementioned forms should be treated as an error.

**Help**

The help command should print out a synopsis of available commands along with a brief description as to what each command does.

**Add**

The add command requires at least 1 additional argument which is a quoted string. The quoting is necessary for your shell to hand off the entire string as a single argument to the script (otherwise, each word would be a separate argument). Note that when passing in a quoted string, the quotes automatically disappear from the input.

An add command should insert and save the specified task into the todo list data file (specified below). If that task already exists in the list (i.e. task description is the same), then an error should be presented to the user. When a task is added to the list, it should be assigned a unique “1-up” integer value (starting with 1 for the first task).

**List**

The list command should display the tasks in the todo list. The output should be neatly formatted (spacing) and should contain the task number, the task description & the due date if specified.

**Delete**

The del command should accept an additional argument which should be an integer greater than zero. The task with this number should then be located, removed and the data file updated accordingly.

There is the possibility for several errors here — the task may not exist, or the argument specified by the user may not be a positive integer. All of these cases should result in an appropriate error message.

**Data File**

The data for this problem **must be stored under a directory called “/todo” in the user's home directory**. **Your script should gracefully handle the case when ~/todo/ doesn't yet exist**. If the user tries to perform any command and it doesn't yet exist, you'll need to handle that gracefully and create directories/files as needed. The format of this file is left completely open for you to decide.

**You are not permitted to write to any location other than the ~/todo/ directory.** If the user wishes to blow out the entire todo list, they may simply remove the ~/todo/ directory and start over again from scratch.

Sample Input/Output

|  |  |  |  |
| --- | --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | | $ ./todo.sh add "TPS Reports"  ADDED #1 - TPS Reports  $ ./todo.sh add "Go fishin' this weekend with Lawrence" "10-03-2020"  ADDED #2 - Go fishin' this weekend with Lawrence  $ ./todo.sh add "Meeting with the Bobs" "22-03-2020"  ADDED #3 - Meeting with the Bobs  $ ./todo.sh add "Help Samir with paper jam"  ADDED #4 - Help Samir with paper jam  $ ./todo.sh add "Update bank software for the 3000 switch" "6/30/2013"  ADDED #5 - Update bank software for the 3000 switch  $ ./todo.sh ***list***  UPCOMING ITEMS:  1: TPS Reports  2: Go fishin' this weekend with Lawrence 10-03-2020  3: Meeting with the Bobs 22-03-2020  4: Help Samir with paper jam  $ ./todo.sh ***del*** 4  DELETED #4 - Help Samir with paper jam  $ ./todo.sh ***list***  UPCOMING ITEMS:  1: TPS Reports  2: Go fishin' this weekend with Lawrence 10-03-2020  3: Meeting with the Bobs 22-03-2020  $ ./todo.sh help  Your clever usage statement here | |
|  | Exercise 2 Write a ***single command*** that will list all the files in your home directory and display the ***number of readable directories*** in it.  You should use the ls, grep and piping for this with the appropriate options. | |

# Exercise 3

1. The file Annotations.txt is a very large file with thousands of records. So don’t try to use cat to display it.

Use the head command with the appropriate parameter to print only the ***first line of the file***.

Note the number/index of the column “code”

This column contains the codes/IDs of different files some of which contain the symbol “\_” in them.

Use the awk command to print out the contents of the column “code” only and calculate the number of codes with the “\_” symbol in them. You can use grep, wc and piping for that with the appropriate options.

1. Use the sed command to delete all the lines that have the symbol “\_” in them ***and keep a backup*** copy of the initial file.
2. Write a bash script that takes one argument, the path to the filtered file ie without symbol “\_”, and that performs the following:
   1. Print out the total number of entries/rows/lines in the file ***excluding the title row***.
   2. Ask the user to enter the name of a new directory to create, ex “filtered\_data”. It should alert the user if the directory already exists so that he/she can enter again a new name without exiting the script. The command ***mkdir*** can be used to create a directory.
   3. Once the directory is created, prompt the user to enter a keyword search: examples to try are “Homo sapiens” and “Mus musculus”

Make sure the search for these keywords is **CASE INSENSITIVE**

* 1. Loop through all the lines in the file and if the line contains the keyword, **CASE INSENSITIVE,** you should write the whole line to a new file. The new file should contain the title row (first line) followed by all the lines that match your keyword and it should be saved in the directory that was created. The name of the file should be the same as the keyword but should not **include any space in it**. For instance if you try “Homo sapiens”, the file name should Homo\_sapiens.txt.
  2. Display the number of lines in the new created file.

# 