Scripting Project 2 – Due 4/5 11:59 pm CST

THIS IS AN EXAM. Any collaboration between students will result in academic integrity charges for all participating students. You may only use the code that I've covered in class to date and your code structure must match mine. You may not seek any outside help. The sample output video is the definitive word on what to do. Your script must run like mine and you must follow my logic. Remember that your script must run to be graded, so it's better to fully finish a function rather than halfway complete them all. You have done everything required for this project in previous tutorials and homework exercises.

Late projects will not be graded. Any script with a modification date later than the due date and time will be considered late.

There is enough flexibility in these instructions that no two scripts should be identical. Using commands not taught in this class will result in penalties. Code that is wildly different than what has been introduced in class will be assumed to be a product of AI generation and will receive no credit.

Cleanly comment your code and use white space to break up blocks of code into logical, commented sections.

Download aggregate_complaints_001.tar.gz and aggregate_complaints_002.tar.gz from the sftp server. I found it handy to make a copy of these two files in a file on my desktop. That allowed me to replace them on the desktop as needed.

CRITICAL: On Thursday, 4/6, you must download the file aggregate_complaints_003.tar.gz from the sftp server. Your home directory should have all three complaint zip folders along with your script clearly named "Project2.sh". It doesn't matter if your Complaints folder is on the desktop or not. Failure to have this exact setup by 4/6 will result in a 10% grade penalty.

We are translating my Python Project 1 solution into BASH. As you look at the solution for Project 1 keep in mind that we are not doing a line by line translation, but rather a process to process translation. I'll give you some top-down guidance here:

```
Shebang

Declarations

Move to home directory

Setup function:
    Check for the complaints folder.  If it is there, remove it then make the necessary folders.  If it is not there just make the necessary folders.  Use static variable for all paths.
    Set up the csv file with the header

Process complaint files:
    Make an array of all zip files to process
    Unpack each tar.gz file then move it to the zip archive.  Files inside each tar.gz go into the Processing folder.
    Notify the user of a wait
    Populate arrays of file names in processing and in file archive
    Ensure that we don't process a file that has previously been processed
    *pay attention to file names.  If I make an array jim of files in the nita directory it would look like jim=("NITA"/*), where NITA is a static variable like NITA="Nita", if that directory was in my home folder.  Each of those file names has the path and the filename!  Pay attention to that, because sometimes you want that and sometimes you will want to cut that string to get just the filename.  This is important.
    Process all file in processing and move each to the file archives as they are processed
    Iterate through the list of files to process:
        Read a file
            If the length of the line = 2 then continue
        Translate the line to replace spaces with underscores
        Make an array by cutting the line (remember the output delimiter)
        Populate variables for each data point (complaint id, date received, company, product and issue)
        Echo these out to the csv file.  Remember to append.
        *NOTE:  you can't toss a READ in the middle of a file read to pause things.  If you want to pause during the file read use this:  read < /dev/tty
        Move the processed file
```

```
Cleanup function:
    Load an array of all file records.  Skip the first line (I just looked for a match on the first line)
    Assign the length of that array to a variable.  That's the number of current records.
    Remove duplicates from that array (sort -u)
    Assign the length of that array to a variable.  That's the number of records after removing duplicates.  Math gets you the number of records removed.
    Destructively write the csv header.
    Iterate through the array and write the lines back to the csv file (append).

Reporting function:
    Read the csv file, skipping the header, to make an array.
    Iterate through that array to make an array of the products
    In a While true:
        Initialize arrays for issue and companies
        Echo the available products header
        Sort the product list to remove duplicates
        List the products by iterating through the product list
            Tr the underscores to spaces
            I had to sed double spaces to single spaces
            Use if to make the product numbers line up like I did
        Get the user product number choice
        Only accept valid product numbers!
        If the product number is zero go back to the main menu
        Get the proper product from the product list, based on the user input
            Assign that to a variable
        Iterate through the big array of all file records.  Cut out the products.  If the product matches the user choice product (the beginning of) then add issues to the issue array and companies to the company array.
        Make a variable equal to the length of the issue array.  That's your number of matches.
        Remove duplicates from the issue array.
        Remove issues from the company array
        Echo the report elements per the demo.  Tr the underscores to spaces.

Call the setup method

Menu: If, three elifs, one else.  Menu choices only refer directly to methods, break, or warn the user of an input error.  Menu is in a While true loop.
```

Is this project error-free?

I sincerely hope so. But of course not. Corrections, changes, etc. will be posted to the Project 2 discussion board. I highly suggest that you subscribe. You are responsible for anything that I post in that discussion forum. If it requires a change in your code then yes: You have to change your code.

Good luck and have fun! It's been a fun project to create.

NOTES:

Mkdir -p to make nested directories like jim/dog

Put quotes around your path variables when making the directories:
Mkdir -p "$THINGS_AND_STUFF"

Do NOT put quotes around your array. If you do, though, use @ instead of # or it will unpack incorrectly.

You can stop the middle of a file read with read< /dev/tty

Processing 1 tar.gz file (the first one) gave me a csv file size of 476.8 kb and 3680 lines. 475.6kb/3762 lines after cleaning.

Processing both tar.gz files gave me a csv file size of 873.4kb/6765 lines. Post-cleaning 868kb/6722 lines.