# Sed

With the aim to fetch student ids and names, a certain someone messed up and

got the data merged instead of separated by commas. So instead of having columns

Student ID, First Name, Middle Name, Last Name

the file just contains

Student ID

with all the entries of a single row merged.

Task : Recover the file back to the format:

Student ID, First Name, Middle Name, Last Name.

You can assume "Student id" can only be numeric and any name starts with

a capital letter. Assume no other special characters. Separate the fields using commas in the ouput.

Input ->

Student ID

1001AaravRohitPatel

1002IshaniRaniSharma

Output ->

Student ID,First Name,Middle Name,Last Name

1001,Aarav,Rohit,Patel

1002,Ishani,Rani,Sharma

Usage :

sed -f names.sed students.csv

# Awk

Study the format of the file "sample.txt" which stores information about various user

accounts. It includes various dummy user accounts too: these have user ids, but they are

not allowed to login. All other users having something other than “/bin/false” as login

shell are valid users and are allowed to login. This is indicated by the setting

"/bin/false" for the login shell.

Task : Write a bash script, "script.sh", using awk to print the user ids (the first field) of all dummy

accounts whose shell is "/bin/false".

The Sample Input File is -

UserID;Name;Designation;loginShell

10001;ajay;manager;/bin/false

1001;sunil;clerk;/bin/false

101;varun;manager;Valid

60123;amit;manager;/bin/false

401;tarun;peon;Valid

And after running the script the output should be -

10001

1001

60123

Note : The input field separator is ";".

Usage :

./script.sh > output.txt

HTML-CSS-JS

------------------- Activity 4 (HTML/CSS/JS) ---------------------

General Details

THEME: Primality Tester

MAXIMUM MARKS: 15 (13 public + 2 hidden)

NOTE: Please ensure that HTML tags and id selectors are correct for auto-grading.

Apart from the mentioned ids, you can use any other ids for your own purpose.

For the image given in the public\_html folder (myimage.jpeg), you can modify that

and add your own image, but make sure to keep the SAME NAME (myimage.jpeg) else

VLab will not pick up the image.

Regarding the screenshots: the primality page has multiple images to show the

working of the function.

NOTE:

1. This activity can be completed by adding all the relevant code in between the start and end

that have been indicated using comments in all the files.

2. Some template code has been filled in. Do NOT add code outside the start and end regions.

3. If you want to use square root function in JavaScript, you can use "Math.sqrt".

--------------------------------------------------------------------------

Editor Details

1. It is recommended to use Visual Studio Code (VSC) for this activity.

2. Once you have fetched the lab files into VLab, minimize the application

3. To access the lab files from VSC, navigate to the BodhiTree workspace and

go to the following path (assuming . is your workspace):

./11/91/264/labDirectory

4. Once you have opened this folder on your local machine, make whatever changes

you want from VSC (or your preferred editor) and then open the HTML files as preview

from within VSC or from the browser in order to see it render

5. MAKE SURE THAT NO FILE IS OPEN IN BOTH VLAB EDITOR AND LOCAL EDITOR AT THE SAME

TIME, ONLY OPEN THE FILE IN ANY ONE EDITOR AT A TIME

6. In case you open the file in both editors by mistake, when you close the file

in VLab, the changes made in local editor will be overwritten, but they can be

restored by "Undo" (Ctrl-Z)

7. You can freely use the evaluator on VLab as many times as you want, but DO NOT

use the VLab file editor to avoid confusion

8. Once you are done editing the files and previewing, and are ready to submit,

open VLab, create a commit, and submit as usual, you can cross-check that all

files are there in VLab editor

9. The only issue that can arise is that changes made in local editor can be lost

due to VLab editor, but follow step 6 to recover it

--------------------------------------------------------------------------

Section A: HTML (4)

The following will require use of HTML. Make sure to check with Sections B and C

to fill up the additional styling/scripting features using CSS/JS. The theme of this

activity is 'Primality Tester' and you will be creating 2 HTML pages: one corresponding

to 'Home' and other (titled 'Prime') will be having a field for the primality tester.

NOTE: the class and ids (wherever they are indicated) are necessary for the autograding.

1. Create a navigation bar (div tag, class=topnav) that redirects to respective pages,

and the pages can either open in the same tab or redirect to a new tab.

You can refer to index.png in the screenshots. (1)

2. Give a 'title' to each page - Home, Prime. Use the same titles in the navigation bar. (1)

3. Home page (index.html): (2)

(a) Add a brief description (p tag, id=desc) of yourself in minimum 150 words (1)

(b) Embed the image (myimage.jpeg, id=myimage), keep its width as 100 and give your name as

title. (1)

--------------------------------------------------------------------------

Section B: CSS (3)

The following will require CSS. The in-line and internal ones need to be added

within respective HTML files, and the external ones need to be added to style.css.

1. Use in-line CSS to: (1)

(a) Make font size of your description/introduction of the Home page 11pt

2. Use internal CSS for: (1)

(a) Styling the navigation bar (use the class to refer to it, you are free to

choose the type of styling)

3. Use external CSS for: (1)

(a) Styling the input field education page (use the HTML tag to refer to it,

style the border to make it as close to the screenshot as possible)

--------------------------------------------------------------------------

Section C: JavaScript (8)

You will need to write the Prime page using a combination of HTML and JavaScript. Create

functions appropriately within HTML files or in script.js as indicated.

NOTE: isPrime(num) should return 0 if num is not a prime and 1 if prime.

1. Prime page (prime.html) (6)

(a) Have a form with a text field 'Enter a number:' and a submit button (2)

(b) Handle the form submission using a function "runFunc" (add this function within

the HTML file itself) (1)

(c) Within the runFunc function, access the number entered in the form, and execute

the function "isPrime" (add this function in script.js) with this number as argument (2)

(d) Take the output of isPrime, and accordingly display a popup that tells whether the

number is a prime or not by saying 'Not a prime number' or 'Prime number' (1)

2. Testcases

The correctness of the isPrime function in script.js will be checked using hidden

input testcases. (2)

Bash

Write a bash script to delete all the files having the extension .py in a given directory, and log all these deletions in a file “log.txt” in the **same directory as the bash script (NOT the directory where the .py files are to be deleted).** Finally print the number of files deleted **to the console (NOT to log.txt)**.

The directory path will be given as an argument to the bash script. Along with deletion of all files, the script should also add the line :- “Deleted <file\_name>” (without quotes) to log.txt.

Usage :- bash submission.sh <path-to-directory>

Eg.

Consider the directory “/home/labDirectory/testfolder” containing files “a.py”, “b.py”, “c.py”,”a.pdf” (quote for emphasis only).

On running the command :- “ bash submission.sh /home/labDirectory/testfolder/”

(inverted commas are not part of the command and meant only for emphasis),

all the .py files in the testfolder directory should be deleted. The only file remaining in the testfolder directory should be “a.pdf” as it does not have a .py extension.

The log.txt file (created in the same directory as submission.sh) should have the following lines:-

Deleted a.py

Deleted b.py

Deleted c.py

Log each individual file on a new line. Be sure to log ONLY the filename and NOT the whole address of the file (i.e. Do NOT write logs of the form “Deleted /home/labDirectory/testfolder/a.py”).

Hint - The basename command might be useful here, use the man pages to see what it does.

You can log the files in any order. Assume log.txt does not exist before running the command.

The terminal should have the output :-

3

Because 3 files are deleted.

Tip - When locally testing, add the delete command when you are sure that the log.txt and count of files is working properly. Else, you will have the hassle of recreating the deleted files after each local test.

Git

A friend of yours tried to merge two branches (master branch, where the HEAD was, and the development branch) of a git repository but couldn’t resolve the merge conflicts in a file. Unfortunately they got too frustrated trying to resolve the errors and finally created a new repository with only the single file (now containing the merge conflict headers added by Git). They bring this new repo to you, and now ask you to help them.

You are given a git repository with a single branch (master) and a single commit, containing a file (file.cpp) with the merge conflict headers added by git. Your aim is to recreate the original 2 files from this file using the merge conflict headers.

After recreating the files, name the file that is identical to the file originally in the master branch (of the original repo) as “file\_master.cpp” and name the file that is identical to the file in the development branch (of the original repo) as “file\_development.cpp”. The original file.cpp should not be present anymore.

Make another commit (that contains only the 2 newly created files) with the message “Recreated files.” (Make sure that **this part of the task is done in only one commit**)

Now, from this commit, create a branch named “development” and in this branch, make a commit with “file\_master.cpp” removed, with the commit message - “Removed file\_master.cpp”.

In the original master branch, make a commit with “file\_development.cpp” removed, with the commit message - “Removed file\_development.cpp”.

The correctness of the files will be judged on the functionality of the code, so don’t worry about matching the original file exactly. As long as the code works as expected, you will be awarded full marks.

“Problem here” - - - -> “Recreated Files” - - - - > “Removed file\_development.cpp” (master)

\

* - - - -> “Removed file\_master.cpp” (development)

The above is a basic structure of what the final commit history should look like.

Effectively, this exercise makes a repository similar to what your friend had accidentally deleted.

Grading -

You have been provided with a Client Side Evaluation Script that checks for the correct repository structure, presence of correct file names, correct commit messages and branch structure. It does not check if the files are compiling and functioning correctly or not (that is done on the server and results are hidden).

Please ensure that you are getting full credit in this evaluation before submitting. If you receive less than the maximum marks in this part, the server evaluation scripts will not be run, and you will be awarded only partial credit (as shown in VLab) for the lab.