****

SED and AWK Problem Statement

****

Solving Instructions for InLab assignment

Strictly follow the instructions given below:

1. Solution for each assignment is to be typed into the .sh or .sed or .awk file already kept inside the corresponding folder.

2. Remember, you need to change the permission of the .sh file before executing it:
chmod u+x <scriptfile> or chmod 777 <scriptfile>

3. Do not change the structure and the names.. The automatic checker will give you a “notsubmitted” grade otherwise, if it does not find the names.

4. Remember to follow the exact output formats specified. Extra characters will lead to incorrect evaluation by automatic checker.

5. See the sample_input and sample_output for actual folder which is given in testcase directory of each problem.

5. Inside the script if you use any temporary file, write command to remove it after the operation within that script only.

6. To check your solution run the script test.sh given in corresponding folder.

Submission Guidelines for assignment **(On moodle)**

---

1. Fill in readme.txt in team-name directory, which contains contribution of each team member and references (cite where you get code/code snippets from).

2. Compress the directory to <team_name>.tar.gz

3. Submit one assignment per team, preferably the lowest roll number.

****

In-lab Assignment

Problem 1:

You are a cracker (an unethical hacker is called a ‘cracker’) and you plan to mint money from a wealthy organization by blackmailing them. You happen to come across a scandalous document which proves that many employees of that organization were involved in a sham that occurred a few years ago.

You are going to ask for a ransom in exchange for keeping the document secret. To start off, you must extract all the email IDs mentioned in the document for sending a threatening message.

So, write a shell script ‘script1.sh’ that will extract all the valid email IDs from a given text file and print them on the terminal (one email ID in one line).

Email id Format: {contains only letters,digits, ‘.’ and ‘_’ }@{letters and ‘.’ only}
Rules for an email address to be valid:

1. It should be in the format mentioned above.
2. There should be at least one character on either side of ‘@’
3. There should be at least one dot character after ‘@’ character.
4. It should definitely end with a letter

Some valid email IDs are given below:

champak@xyz.com
supandi@yahoo.co.in
tumtum@cse.iitb.ac.in
t1umtum@.cse.iitb.ac.in
huk_um.chalisa123@gmail.com

Invalid email IDs :
dha#csg@gmail.com
tumtum@cse.
Rham.hsdfk.234.456

The script output will be graded programmatically, and any idiosyncrasies shall lead to deduction of marks.

How to execute:
./script1.sh <inputFileName>
(here filename is passed as an ARGUMENT and NOT AS AN INPUT STREAM)

NOTE: The email addresses should be printed in the same order as their order of occurrence in the input file.

Problem 2:

Your computer networks project deadline is knocking on the doors and you have a considerable amount of work left. A major portion of the project requires you to validate IP addresses and map the valid IP addresses to their respective classes. I am sure you can easily find out on your own what a valid IP address is.

You have a text file of a log that contains some addresses and you want to write a script that will automate this task. [Input]

Write a shell script that will extract valid IP addresses from a given text. For every valid IP address found, print its corresponding class.

You should write your script in 'script2.sh' file provided inside the directory. If there are 'n' valid IP addresses in your script, then your output should be 'n' IP addresses followed by their class separated by a space.

IPv4 Format:
w.x.y.z , where w,x,y,z are in range [0,255].
Class A : w is in range [0,127]
Class B : w is in range [128,191]
Class C : w is in range [192,223]
Class D : w is in range [224,239]
Class E : w is in range [240,247]
Not Defined : w is in range [248,255]

How to execute:
./script2.sh <inputFileName>
(here filename is passed as an ARGUMENT and NOT AS AN INPUT STREAM)

Print the output on the terminal itself.
e.g., if the given text contains 6 valid IP addresses, then your output should look like:

10.12.144.36 A
178.152.36.45 B
192.168.12.1 C
232.56.23.1 D
245.23.123.234 E
250.2.3.45 Not Defined

NOTE: The ip addresses should be printed in the same order as their order of occurrence in the input file.

Problem 3:

Your friend has completed his CS101 assignment and you feel numb that you could not even started it. In the end, you are just bothered with marks and simply want to come crawling out of the course.
After getting a papercut you have decided to get it from him over e-mail, and not write it down.
You are dead from the inside and breaking the habit of being able to copy an assignment and/or thinking what i've done after every submission, is getting harder day by day.

The first step to copying an assignment is ensuring you do not get caught when you submit it; the little things give you away.

You want to remove all the comments from his assignment to get one step closer to the submission. [We won't mind if the readme of this assignment contains the name of "you-know-what" we are referring to in the text above" ;)]

Strong disclaimer: We do not encourage you to copy assignments at all. In the words of Arnab Goswami - "Never ever! Ever Ever!" The consequences of getting caught in this department are pretty rough. You do not believe me? Ask your seniors what a D-ADAC is!

Write a SED script "script3.sed" which cleans both kinds of comments from the C file (// and /* */ - single line and multi line) and prints the cleaned code on the terminal (no printing in a separate file; no output file to be generated).

The script output in the input file itself will be graded programmatically, and any idiosyncrasies shall lead to deduction of marks.

Input :
C program file name
sed -f script3.sed <inputFile>
(here filename is passed as an ARGUMENT and NOT AS AN INPUT STREAM)

Output:
Cleaned C program on the terminal.

NOTE: Cleaning here means that the comments should be deleted i.e replace them with nothing. Don't replace them with spaces or newlines.

Problem 4:

A program written by you gets the input from an embedded systems device (Joystick on a gamepad) through a serial port on your machine. The program generates output in the following format:

1,X!2,Y!3,Z!2.5,O!

where the , denotes a field separator and ! denotes a record separator.

Write an awk program script4.awk which takes as an input argument a file of the format specified above and generates the output on command line (prints the output; not in a file, for the n-th time, not in a file) of this format:

How to Execute:
awk -f script4.awk <inputFile>
(here filename is passed as an ARGUMENT and NOT AS AN INPUT STREAM)

Output:

Value	SensorNumber
1	X
2	Y
3	Z
2.5	O

Note that the columns are tab separated.

The above should be the output on the command line and nothing else should be generated, not a file, not anything extra written. The script output will be graded programmatically, and any idiosyncrasies shall lead to deduction of marks.

GIT SUBMISSION INSTRUCTIONS (Only for OutLab, NOT InLab)

1. Create a repository with the name <team_name>_inlab3.
2. Only **one member** among the team should create this repository. The one who creates will give access to his team members and the respective TA alloted to the team.(*TA list will be posted on moodle*).
3. Clone this repo into the local lab machine you are working on.
4. Create three new branches(one by each team member) out of this repository.(*Hint: Learn about 'checkout command of git branching'*)
5. Start working in your respective branches and at the end, merge the content in all three branches to the master branch. (*Take care of merge conflicts*)

Reference for git:
https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging