CS 251 Outlab 2 : RegEx (Sed + Awk) - 2019

CS251 Outlab 2 : RegEx (Sed + Awk)

Please refer to the general instructions and submission guidelines at the end of this document before submitting.

(For all the questions, if you might be creating any intermediate file which is not asked to be generated then please put-in appropriate commands to delete those files as a part of your bash scripts itself)

Task 1 - Cryptography ! (Marks)

Prologue

A conspiracy is being planned to assassinate Caesar by his own Senate Members. His only chance of staying alive is to decode '**TFEXIRKLCRKZFEJ WFI KYVV YRKY JRMVU TRVJRI**', which is known to be a form of **Caesar Cipher**. But he doesn't have the expertise to decode it. So now the onus of saving him is on you, for you are known to be the best mathematician in the Roman Republic. It's time for you to prove your valor.

Problem

The Caesar Cipher shifts all the letters in a piece of text by a certain number of places. The key for this cipher is a letter which represents the number of place for the shift. So, for example, a key D (4th letter) means "shift 3 places" and a key M (13th letter) means "shift 12 places". Note that a key A means "do not shift" and a key Z can either mean "shift 25 places" or "shift one place backward".
For example, the word "CAESAR" with a shift P becomes "RPTHPG".
Caesar himself used 3 as a key for protecting messages of military significance however the key to this cipher is unknown and hence Caesar has come to you for help.

Subtasks

A. Write a bash script named **saveCaesar.sh** to find the key of the given cipher. The structure of your script should be as following:

a. Input the cipher to the script as a string (with spaces preserved) using the **read** command

b. Increment the characters of the input string one-by-one and print it on the terminal while maintaining a **key** variable alongside. Look for meaningful words in the stdout output (Hint: use **tr**)

c. stdout for saveCaesar.sh should look like:

A <decoded_text_with_0_shifts>
B <decoded_text_with_1_shift>
C <decoded_text_with_2_shifts>
…….
Z <decoded_text_with_25_shifts>

d. Note down the words (by typing in yourself) which you find meaningful and the corresponding key-value and save it in **decodedCipher.txt** in the following format: (a two-line output)
<key>
<decoded_cipher>

B. Having saved Caesar now you plan to retaliate back to decimate your enemies. So write a bash script named **retaliation.sh** for sending a protected message 'KILL ALL' to your military forces which can have a variable key. The script structure is as follows:

a. The script takes the **key** value as a command-line argument
b. The message to be encoded is again taken from stdin using the **read** command (similar to the a. part)
c. The output should simply be the encoded message encrypted using the key value and print that encrypted message to stdout

---

Task 2 - PDF Scraping ! (Marks)

Here is a PDF containing the dummy student data of their

1. Name
2. Roll Number
3. CPI
4. Department
5. Courses Undertaken

Your goal is to write multiple bash scripts to scrape the PDF and extract the required data from it.

(All the files and folders generated from the bash script as a part of these sub-tasks should be created in the same directory as the script which creates them e.g. summaryData.txt should be located in Task2/A/)

Subtasks

A. Write a bash script named **pdfScraper.sh** (Usage: **./pdfScraper.sh <URL>**), which

a. Takes the URL mentioned as the input; then
b. Downloads the PDF; then
c. Converts the PDF file into a text file and saves it by the name **summaryData.txt** in the same folder as where the script is located (Hint: use **pdftotext**); then
d. Deletes the PDF file that was generated

B. Write a bash script named **csvGenerator.sh** (Usage: **./csvGenerator.sh ../A/studentData.txt**), which

a. Takes **studentData.txt** as the input; then
b. Generates a CSV file **studentData.csv** , from the input, with Student Name, Roll Number, CPI, Department, Courses Undertaken as the fields in it and "|" as the delimiter. i.e the contents of the CSV file should be in the following format:

Student Name|Roll Number|CPI|Department|Courses Undertaken
<Student_Name>|<Roll_Number>|<CPI>|<Department>|<Courses_Undertaken>
<Student_Name>|<Roll_Number>|<CPI>|<Department>|<Courses_Undertaken>
……..
<Student_Name>|<Roll_Number>|<CPI>|<Department>|<Courses_Undertaken>

C. Write a bash script named **sortStudentData.sh** (Usage **./sortStudentData.sh ../B/studentData.csv**), which

a. Takes **studentData.csv** as the input; then
b. Sorts the CSV file in descending order and stored the result in another CSV file named **sortedStudentData.csv** ; then
c. Stores the names of the top 5 students with maximum CPI in a text file named

top5Students.txt such that every name is in a new line

D. Write a bash script named **studentDataExtractor.sh** (Usage: **./studentDataExtractor.sh ../A/studentData.txt**), which

a. Takes **studentData.txt** as the input; then

b. Generates independent text files named **<Roll_Number>.txt** for each and every student containing all the data pertaining to that particular student. This link might help you solve it.

(i.e. simply copy-paste all the text pertaining to a specific student to the respective text file, but, do not copy the '……' line to the output text files)

E. Write a bash script named **departmentFilter.sh** (Usage: **./departmentFilter.sh ../B/**), which

a. Takes the path of the folder containing all the individual student data text files (i.e. **../B/**) as input; then

b. Filters each of these text files into different folders based on the department of the student. i.e. all the text files of the students in the Civil Engineering Department should be stored in a folder named **CivilEngineering** . Similarly, folders named **ComputerScienceAndEngineering** and **ElectricalEngineering** for the text files of the students belonging to the respective departments

(Note: Do not hard code the initial generation of these empty folders named by the department names since the hidden test cases might have some other department names)

Task 3 - COAT(Course Organizer and Analyser in Terminal)(marks)

Aim: It is always a good habit to plan ahead. This task will help you to create a course visualizer in your terminal and find out how many credits are still left for your

Resources/Inputs:

1) a file containing the list of courses taken by an individual user is given as a CSV file where its first line is having the columns course-code, semester, year, credits, letter grade stored in CSV format .(**allCoursesTaken.csv**)

2) Requirements of no. of credits under an exhaustive list of tags available. It also has color codes mapped to corresponding tags. This information is lifted from here.(you will use it to glorify the boring terminal). (**creditsRequirements.csv**)

3) association of grade with the grade point is given in the file.(**letterGradeToNumber.csv**)

Subtasks

A. Create an awk file **viewWithoutColor.awk** which displays the data in allCoursesTaken.csv in a formatted way in the terminal. See sample output to get what does formatting means. Note you need to remove the column "Name" and keep the width of each field to be 20 columns (use printf("%20s",$3)). Note the number of highens for the header is 20*(number_of_fields). This should be a generic script and should work on all kinds of tables. Only things hardcoded will be 20 and Name.

Command :- **awk -f viewWithoutColor.awk allCoursesTaken.csv**

From here onwards output of task a will be named **output**

B. Create a script **viewWithColor.sh** which uses the outputA as input along with the creditsRequirement.csv and uses sed and awk to colorize the output along the lines of the color scheme mentioned in creditsRequirement.csv . Look at the file defineColors.sh which

is provided to help you in the task. Look at the intended output and do a cat of it to understand what to do. Also, look at this , this link to understand how the coloring actually works.
Command:- **./viewWithColor.sh outputA creditsRequirements.csv**

C. Create a script **viewSemester.sh** that takes in 3 argument . one as the outputA and the other two as the semester and year. Then it sorts the input w.r.t. the course code and display the same.

D. Create a script **viewCourse.sh** that takes as input outputA and another search_string for course_code. You need to display all the courses having the search_string as a substring of the course_code

E. Create a script **calculateSPI.sh** that takes in same input as **viewSemester.sh** and uses viewSemester.sh internally followed by calculating the SPI for that semester (Minor SPI, Major SPI and Honors SPI).

F. Create a script **./calculateCPI.sh** that takes outputA as input, uses the script **calculateSPI.sh** and displays the CPI. (to be updated)

G. Create a script **./viewRemaining.sh** to get a summary of the number of credits done under each tag and the number of credits remaining. (to be updated)

General Instructions

- Make sure you know what you write, you might be asked to explain your code at a later point in time.
- The submission will be graded automatically, so stick to the naming conventions strictly.
- The deadline for this lab is **Sunday, 11th August, 11:55 PM.**

Submission Instructions

After creating your directory, package it into a tarball **`outlab2-<team_name>.tar.gz`**
Submit once only per team from the moodle account of the smallest roll number.
The directory structure should be as follows (nothing more nothing less). Also, even if you don't have any references then please just add an empty text file with the same name.

```
outlab2-<team_name>/
├── Task1
│   ├── A
│   │   ├── saveCaesar.sh
│   │   └── decodedCipher.txt
│   └── B
│       └── retaliation.sh
├── Task2
│   ├── A
│   │   └── pdfScraper.sh
```

```
│   ├── B
│   │   └── csvGenerator.sh
│   ├── C
│   │   └── sortStudentData.sh
│   ├── D
│   │   └── studentDataExtractor.sh
│   └── E
│       └── departmentFilter.sh
├── Task3
│   ├── A
│   │   └── viewWithoutColor.awk
│   ├── B
│   │   └── viewWithColor.sh
│   ├── C
│   │   └── viewSemester.sh
│   ├── D
│   │   └── viewCourse.sh
│   └── E
│       └── calculateSPI.sh
└── references.txt
```