System Commands

OPPE 2 Set 1 and Set 2 Combined

---

24 July 2022 [SET 1]

Section 1 [Attempt any 1]

Question 1 [15 mark]

Write a **sed script** to

- Swap the first and second fields in the given input having field separator as a colon `:`.
- Replace every occurrence of the character `?` if found at the end of a line to `!`.

Note: complete partial tasks for partial marking.

Test case description: The input is the input file. And output is the output of running your sed script.

Solution

```
s/^\([^:]*\):\([^:]*\)/\2:\1/
s/?$/!/
```

Question 2 [15 marks]

Write a **sed script** to

- Replace every three letter abbreviation of day of week to the respective number as given in the table. The abbreviation can be in any case or mixed case

Abbreviation(case insensitive match)	Replace with
sun	1
mon	2
tue	3
wed	4
thu	5
fri	6
Sat	7

- Change the first and third occurrences of the character `!` to `.`, on each line.

Note: complete partial tasks for partial marking.

Test case description: Input is the input file to sed. Output is the output printed on running your sed script.

Solution

```
s/sun/1/gI
s/mon/2/gI
s/tue/3/gI
s/wed/4/gI
s/thu/5/gI
s/fri/6/gI
s/sat/7/gI
s/!/./1
s/!/./2
```

Section 2[Attempt any 1]

Question 1 [20 marks]

Given a file that contains current years board exam scores of students in all the schools in an area. Each line in the file contains four **comma-separated** fields: school code, roll number of the student, name of the student, and marks in the below format.

```
School_code,Student_Roll_no,Name,Marks
```

All the fields could be alphanumeric values except the last field `Marks` which is a number less than 500, as the maximum marks of the exam is out of 500.

Write an **AWK script** to print the roll numbers of the toppers of each school. For example if there is marks details of students of 11 schools of an area in the file, then your output should contain 11 roll numbers in any order, one for each school.

Test case description: Input is the contents of the input file. Your script does not need to read any input. Output is the expected output from your script. Your script may print expected roll numbers in any order, evaluation script sorts the input before printing.

Solution

```
BEGIN {
  FS=","
}
{
  if ($4 > max[$1]) {
    max[$1] = $4
    max_student[$1] = $2
  }
}
END {
  for (i in max_student) {
    print max_student[i]
  }
}
```

Question 2 [20 marks]

Write an **AWK script** to print the number of user defined functions and the number of calls made to these functions in an input python program file. Functions may or may not contain arguments.

The output consists of two parts

- First line of the output is a number that is the number of user defined functions in the input file.
- Followed by one line for each user defined function counted above, print the number of calls to each function in the format `<function_name>:<number_of_times_called>`. Just print the function name, not the arguments or brackets, and in the same order they appear in the input file.

E.g. consider the below function definitions in the input file.

```
def function_1(argument1="default"):
 print("something");

def function_2():
 print("This is function_2");
```

If `function_1` is called 3 times and `function_2` is called 4 times and these are the only two user defined functions in the input file then the output from your script should be.

```
2
function_1:3
function_2:4
```

In Python a function is defined using the `def` keyword. And is called by using the function name followed by parenthesis. As shown belowExamples`# This is a comment# function definition of function_1()def function_1(): print("this is a function") # function call  function_1()`

Note: Assume that there are no recursive calls, and no user defined function will call other user defined functions. But some test cases may contain string similar to function call in comments that should be ignored for full score.

Test case description: Input is the contents of the input file on which your AWK script will be run. Your script does not need to read any input. Output is the output printed by your script.

Solution

```
/^[^#]/{
  if ($0 ~ /^def [[:alnum:]_]+\(.*\):$/) {
    split($0, t1)
    split(t1[2], t2, "(")
    fn[t2[1]]=0
  }

  else {
    for (f in fn) {
      if ($0 ~ f"\\(.*\\)") {
        fn[f]++
        #print $0":"f"->"fn[f]
      }
    }
  }

}

END {
  for (f in fn) { n++; }
  print n
  for (f in fn) { print f":"fn[f]; }
}
```

Section 3 [Attempt all]

Question 1 [15 marks]

A confidential and long text file has to be shared with others. Before sharing the ownership of the file wants to hide some sensitive text.

Write a **sed script** that will print the file after making the below changes.

- If any line contains any of the strings `Password`, `password`, `Address`, `address`, or the line consists of only digits then the entire line should be replaced with the string `##REDACTED##`. Note that the word `REDACTED` is in capitals with two `#` characters on each side in the replaced string.
- Add the line `##CONFIDENTIAL##` before the first and after the last line of the file. Note that the word `CONFIDENTIAL` is in capitals with two `#` characters on each side, there should be no other characters(including spaces) on this line.

Note: complete partial tasks for partial marking.

Hint: use command like `/pattern/ c string` to replace complete line containing the `pattern` with `string`.

Test case description: The input is the input file. And output is the output of running your sed script.

Solution

```
1 i\##CONFIDENTIAL##
/[Pp]assword/ c ##REDACTED##
/[Aa]ddress/ c ##REDACTED##
s/^[0-9][0-9]*$/##REDACTED##/
$ a\##CONFIDENTIAL##
```

Question 2 [15 marks]

Write an **AWK script** that prints the input file after the following changes:

- If there are exactly two fields in the line(record), swap the first and second column.
- If there are more than two fields in the line then print it as it is.

Take the input field separator for the given input as a space character. Keep the same field separator in the output too.

Note: For partial marks you can submit the script that swaps first and second column on each line irrespective of the number of fields in the line.

Test case description: Input is the contents of the input file on which your script will be run. Your script does not need to read any input. Output is the expected output from your script.

Solution

```
{ if (NF==2) print $2,$1
  else print $0
}
```

Question 3 [15 marks]

You run a bash utility that regularly creates a lot of files in the directory named `outfiles`. One file is duplicated many times among these files. You want to eliminate duplicates and keep only one copy. We do not know which file is duplicated, but only one file is duplicated that we know. Except for these duplicates all others are distinct. Write a **Bash script** that finds all the duplicate files directly under the directory `outfiles` and removes the duplicate files. Among the duplicate files keep the file whose name appears first in lexicographically sorted order. i.e. if files `de`, `dgt`, `we` and `bb` are duplicates, keep the file `bb` and remove the files `de`, `dgt` and `we`.

If required use the bash script named `printdup.sh` in your script, which is located in the current working directory. Run the command `bash printdup.sh outfiles` to print all the duplicate files in the directory `outfiles`. The output of this command will contain file paths of duplicate files with respect to the current working directory.

E.g. if the files `de`, `dgt`, `we` and `bb` present in the directory `outfiles` are duplicates, then the following will be the output if run under the current working directory. Note that the files may or may not be sorted lexicographically in the output of `printdup`.

```
$ bash printdup.sh outfiles
outfiles/de
outfiles/dgt
outfiles/we
outfiles/bb
```

Note: Do not add/remove/edit any files other than the ones specified in the problem.

Hint: On how to read a file/input line by line in bash, refer cheatsheet. Alternative to reading line by line, `xargs` can be used.

Test case description: Input is read by the evaluation script, your script does not need to read any input. First line of input is the list of all duplicate files in the directory `outfiles` and second line of input is the list of all remaining files in the directory `outfiles`. Your script does not need to print anything. Expected output is the list of files in the directory `outfiles`, printed by evaluation script after running your script.

Solution

```
bash printdup.sh outFiles | sort >files.dup

flag=0
while read line; do
  if [[ "$flag" == "0" ]]; then
    flag=1;
    continue;
  fi
  rm $line
done <files.dup

# Alternate solution
# bash printdup.sh outfiles | sort | sed -n '1!p' | xargs rm
```

Question 4 [20 marks]

Consider the network log file named `network.log.0` located in the current working directory, in which all the incoming logs will be appended. This file can grow in size easily so we want to rotate these log files if the size of the files increases beyond a limit. Write a **Bash script** that rotates the log file `network.log.0` if its size is greater than 200 bytes as described below.

- If the size is greater than 200 bytes, then
  - Rename the file to `network.log.1`, if `network.log.1` already exists in the current working directory then rename the file to `network.log.2`, if `network.log.2` also exists then rename the file to `network.log.3` and so on.
  - Create a new empty file named `network.log.0` to be used for new incoming logs.
  - And print `ROTATED` on stdin.
- If the size is less than or equal to 200 bytes print the size of the log file `network.log.0` to stdout. E.g. if the size of file in bytes is '124' then output should just be a number `124'.

Note: The log files are always rotated using your Bash script, so the rotation will always be in sequence. i.e. `network.log.3` exists, then `network.log.1`. `network.log.2` and `network.log.3` will always exist in the current working directory. You need to find the highest numbered log file and rotate the log based on that.

Hint 1: Bash command `du -b <fileName>` will give the size of the file `<fileName>` in bytes. The output will contain tab-separated values, the first value will be size in bytes second value will be a filename. Also, note that the `cut` command's default delimiter is tab character. Hint 2: If required use the command `sort -n` for numerical sorting.

Example `du` output

```
$ du -b network_log
167 network_log
```

Test case description: First line of the input contains numbers used by evaluation script. Your script does not need to read any input. Output is the output printed by your script if the task is accomplished successfully.

Solution

```
MAX=200
filename=network.log
size=`du -b $filename.0 | tr -s '\t' ' ' | cut -d' ' -f1`
if [ $size -gt $MAX ]
then
    # The below logic will work only if log files are less than 10
    # count=`ls $filename* | tail -1 | cut -d'.' -f3`

    # So.
    count=`ls $filename* | cut -d'.' -f3 | sort -n | tail -1`

    count=$((count+1))
    mv $filename.0 $filename.$count
    touch $filename.0
    echo "ROTATED"
else
    echo "$size"
fi
```

---

31 July 2022 [SET 2]

Section 1 [Attempt any 1]

Question [15 marks]

In a programming course, the project vivas are scheduled in the month of July and August. Due to some issues, every scheduled viva should be postponed exactly by a month on the same date. E.g. a viva scheduled on `19 July 2022` should be rescheduled to `19 August 2022` and the viva scheduled on `6 August 2022` should be rescheduled to `6 September 2022`. Note that there are 31 days in July and August but only 30 days in September. So vivas scheduled on `31 August 2022` should be rescheduled to `1 October 2022`.

The viva details are noted in a file. Each line in this file contains a viva schedule for one student, check the public test case for the details on the format. The months are written in full in this file(like August and not Aug) but can be written with a mix of capital and small letters(like August or AUGUST or august). Also, note that every string in this file is separated by a space.

Write a sed script that works on the viva file as input and does all the replacements to reschedule the vivas as given above. After replacement, the month strings should be in all capitals like `AUGUST`, `SEPTEMBER` and `OCTOBER` in the output.

Hint: You need to pay attention to the order of replacements.

Test case description: Input is the input file to your sed script. Output is the output printed on running your sed script. Your script does not need to read input.

Solution

```
s/31 august/1 OCTOBER/gI
s/August/SEPTEMBER/gI
s/july/AUGUST/gI
```

Question [15 marks]

A file needs to be shared with the vendors, but this file contains the PAN number of the customers. Hence we need to replace these PAN numbers with some string before sharing.

Write a sed script to replace all PAN numbers with the string `##HIDDENPAN##`.

Every PAN number is a ten-digit unique alphanumeric string with the following structure `XXXZX1234X`

- The first three characters `XXX` are all English alphabet.
- The fourth character `Z` is an English letter from either of the letters C, P, H, F, A, T, B, L, J, or G.
- The fifth character `X` is an English alphabet.
- The next four characters are sequential numbers running from 0001 to 9999.
- Last character `X` i.e the tenth character is an English alphabet.

Note: All the letters in PAN are either uppercase letters or digits.

Test case description: Input is the input file to your sed script. Output is the output printed on running your sed script. Your script does not need to read input.

Solution

```
s/\b[A-Z]\{3\}[CPHFATBLJG][A-Z][[:digit:]]\{4\}[A-Z]\b/##HIDDENPAN##/g
```

Section 2 [Attempt any 1]

Question [20 marks]

Given a file that contains current years' board exam scores of students in all the schools in an area. Each line in the file contains four **comma-separated** fields: school code, roll number of the student, name of the student, and marks in the below format.

```
School_code,Student_Roll_no,Name,Marks
```

All the fields could be alphanumeric values except the last field `Marks` which is a number less than 500, as the maximum marks of the exam is out of 500.

Write an **AWK script** to print the number of students who have got more than 300 marks in each school in the format `School_code:number_of_students_scored_above_300` For example, if there are marks details of students of 3 schools of an area in the file,

```
A,A70368835,Rose,102
A,A57728910,Alvaro,439
A,A56178246,Harrison,122
C,C91473470,Jacklyn,328
A,A59855154,Keisha,220
C,C3111718,Tanya,300
B,B46578794,Cameron,100
B,B1227361,Anna,192
C,C62475945,Alvaro,452
A,A3422606,Elizabeth,353
```

then your output should contain 3 lines in the output, one for each school. Like:

```
A:2
B:0
C:2
```

Test case description: Input is the contents of the input file. Your script does not need to read any input. Output is the expected output from your script. Your script may print school codes in any order, the evaluation script sorts the output before printing.

Solution

```
BEGIN { FS="," }

!more_than_300[$1] { more_than_300[$1] = 0 }
$4 > 300 { more_than_300[$1]++ }

END {
  for (school in more_than_300) {
    print school":"more_than_300[school]
  }
}
```

Question [20 marks]

Consider a comma-separated file containing details of all the employees of an organization. Each line in the file contains employee Id, department Id, leaves taken this year and the gender of the employee in the format. `employee_ID,department_ID,leaves_taken,gender` `employee_ID` and `department_ID` are alphanumeric values, `leaves_taken` is an integer and `gender` can have only two values `male` or `female`.

Write an **AWK script** to print the minimum number of leaves taken by employees in the organization as described below. In your output:

- The first line should be a number that denotes the minimum number of leaves taken by any employee in the whole organization.
- Followed by `N` lines of output, where `N` is the number of departments denoted by a unique `department_ID`. Each of these `N` lines should be in the format `department_ID:MIN_LEAVES_TAKEN` Where `department_ID` is a unique department Id, `MIN_LEAVES_TAKEN` is the minimum number of leaves taken by any employee in the department denoted by `department_ID`.

Sample Input

```
C89707250,C,13,male
A8825622,A,17,male
C2366741,C,12,male
B67143992,B,15,male
B45126668,B,5,female
```

Sample Output

The below sample output shows that the minimum leaves taken by any employee in the whole organization is `5`. There are three departments in the organization so the following three lines give the minimum leaves of employees in each department.

```
5
A:17
B:5
C:12
```

Note: The number of leaves taken by any employee is less than `31`.

Hint: Use `length(var)==0`, to check is a variable is null or not set.

Test case description: Input is the contents of the input file. Your script does not need to read any input. Output is the expected output from your script.

Solution

```
BEGIN{
FS=",";
}

length(leaves[$2])==0 || $3 < leaves[$2] { leaves[$2] = $3 }
length(tmin)==0 || $3 < tmin { tmin = $3 }

END{
  print tmin
  for (dpt in leaves) {
    print dpt":"leaves[dpt]
  }
}
```

Section 3 [Attempt all]

Question 1 [15 marks]

A file contains only two fields separated by a delimiter which is a combination of 3 space characters. Also after every 5th line a block separator is inserted which is a line containing only the string `####`. Write a **sed script** to

- Replace only the 2nd space in every delimiter with a pipe `|` character.
- Delete the block separator lines. The first block separator is at the 6th line then after every 5 lines.

Sample Input

```
d   31
a   4
x   12
first   second
g   0
####
p   q
r   s
t   12
10   10
3   dup
####
last   1000
```

Sample Output

```
d | 31
a | 4
x | 12
first | second
g | 0
p | q
r | s
t | 12
10 | 10
3 | dup
last | 1000
```

Test case description: The input is the input file. And output is the output of running your sed script.

Solution

```
s/ /|/2
6~6d
```

Question 2 [15 marks]

A student has to submit his data in a text file containing keys on the odd-numbered lines and the corresponding values on the next even-numbered lines, as shown below.

```
key1
value1
key2
value2
key3
value3
```

But the student made a mistake and his file `data.txt` looks like below,

```
value1
key1
value2
key2
value3
key3
```

Write an **AWK script** to print the file `data.txt` after correcting the mistake. Basically, your AWK script should swap every two consecutive lines in the file. Assume that the file will always contain even number of lines.

Test case description: Input is the contents of the input file. Your script does not need to read any input. Output is the expected output printed as it is by your script.

Solution

```
NR % 2 == 1 { valueLine = $0 }
NR % 2 == 0 { print $0; print valueLine }
```

Question 3 [15 marks]

Consider an application which runs several services. Every service dumps logs to current log files and when this log file size increases beyond a certain size the logs are rotated(i.e. new log file are created for dumping upcoming logs and old log files are retained). All the log files are named in the same format. Every service dumps logs to the current log file named `<service-name>.log.X`, where `<service-name>` is a unique string(not containing space or dot) for each service and `X` is the highest number among all the log files. These log files are located in their respective directories located inside the directory named `log` which is located in the current working directory.

An example directory structure is shown below, here there are log files of three services namely `acpid`, `alsa-restore` and `apparmor` located in their respective directories. The current log file for service `acpid` is `log/acpid/acpid.log.4`, similarly for service `armor` the latest log file is `log.armor/armor.log.3`.

```
.
├── log
│   ├── acpid
│   │   ├── acpid.log.0
│   │   ├── acpid.log.1
│   │   ├── acpid.log.2
│   │   ├── acpid.log.3
│   │   └── acpid.log.4
│   ├── alsa-restore
│   │   ├── alsa-restore.log.0
│   │   └── alsa-restore.log.1
│   ├── armor
│   │   ├── armor.log.0
│   │   ├── armor.log.1
│   │   ├── armor.log.2
│   │   └── armor.log.3
```

Write a **bash script** to create symbolic links for each service in the directory `log` pointing to the current log file of each service. All the soft links should be placed in the directory `log` and should be named as `<service-name>.log`. E.g. for the above directory structure, for service name `acpid` the soft link named `acpid.log` should be created in the directory `log` and should be linked to the file `acpid/acpid.log.4`.

After running your script final directory structure should look like.

```
.
├── log
│   ├── acpid.log -> acpid/acpid.log.4
│   ├── alsa-restore.log -> acpid/alsa-restore.log.1
│   ├── armor.log -> armor/armor.log.3
│   ├── acpid
│   │   ├── acpid.log.0
│   │   ├── acpid.log.1
│   │   ├── acpid.log.2
│   │   ├── acpid.log.3
│   │   └── acpid.log.4
│   ├── alsa-restore
│   │   ├── alsa-restore.log.0
│   │   └── alsa-restore.log.1
│   ├── armor
│   │   ├── armor.log.0
│   │   ├── armor.log.1
│   │   ├── armor.log.2
│   │   └── armor.log.3
```

Hint: For corner case default sorting may not be useful. See below sort command man page snippet.

```
sort - sort lines of text files
      -n, --numeric-sort
            compare according to string numerical value
```

Test case description: Input is the set of file path read by the evaluation script. Your script does not need to read any input. Output contains success/failure message based on the tasks completed, your script does not need to print anything.

Solution

```bash
cd log
for pdir in *; do
  # The below logic will work only if log files are less than 10
  # latestCount=`ls $pdir | tail -1 | cut -d'.' -f3`

  # So.
  latestCount=`ls $pdir | cut -d'.' -f3 | sort -n | tail -1`
  ln -s "${pdir}/${pdir##*\/}.log.$latestCount" "${pdir##*\/}.log"
done
```

Question 4 [20 marks]

Write a **Bash script** that accepts two file names as command line arguments, and swaps the contents of these files.

The command line arguments may or may not be equal to two, and the files given may or may not exist or have write permissions on them. In these cases print the error messages to **STDERR** and exit with the exit codes as per the table below.

Condition	Exit code	Error Message
Number of arguments is less than two or greater than two.	1	ARG ERROR
One or both files does not exist.	2	NOT EXISTS
One or both files are not writable.	3	WRITE ERROR

If none of the above conditions is true and the files are swapped successfully, exit with exit code `0` and print `SUCCESS` to STDIN.

Test case description: Input contains file names, their contents and some additional info which is read by evaluation script. Your script does not need to read any input. Output is the expected output that should be printed by your script.

Solution

```
if [[ $# -ne 2 ]]; then
  echo ARG ERROR >&2
  exit 1
fi

if [[ ! -f $1 || ! -f $2 ]];then
  echo NOT EXISTS >&2
  exit 2
fi

if ! [[ -w $1 && -w $2 ]];then
  echo WRITE ERROR >&2
  exit 3
fi

mv $1 tmp
mv $2 $1
mv tmp $2

echo SUCCESS
```