CS108 - Software Systems Lab

Lab 10 - `sed` and `awk`

Student: Aditya Sanapala, 23b0912@iitb.ac.in
Lecturer: Kameswari Chebrolu, chebrolu@cse.iitb.ac.in

Problem 1: `sed` life

You realize that the entries in the file `employment.csv` with `Film` as `High_industry` have to be replaced by `Film&Music`. Being familiar with `sed`, you try out replacing each instance of `Film` with `Film&Music`.

Task: Given the file `employment.csv`, you need to use substitute command of `sed` to replace all instances of `Film` to `Film&Music`.

Usage: `sed -f substitute.sed employment.csv > output.txt`

Example:

Input:

```
2019-05-05,Number of paid jobs - 34 days,Film,2090110
2019-05-05,Number of paid jobs - 34 days,Agriculture,95150
2019-05-05,Number of paid jobs - 34 days,Banking,405050
2019-05-05,Number of paid jobs - 34 days,Manufacturing,1570740
2019-05-05,Number of paid jobs - 34 days,Railways,19170
2019-05-05,Number of paid jobs - 20 days,Film,1828160
2019-05-05,Number of paid jobs - 20 days,Agriculture,79880
```

Output:

```
2019-05-05,Number of paid jobs - 34 days,Film&Music,2090110
2019-05-05,Number of paid jobs - 34 days,Agriculture,95150
2019-05-05,Number of paid jobs - 34 days,Banking,405050
2019-05-05,Number of paid jobs - 34 days,Manufacturing,1570740
2019-05-05,Number of paid jobs - 34 days,Railways,19170
2019-05-05,Number of paid jobs - 20 days,Film&Music,1828160
2019-05-05,Number of paid jobs - 20 days,Agriculture,79880
```

Problem 2: Another day, another `sed`.

After quickly finishing your C++ script, you realize that instead of `pprint` (the function that you have defined) you have written `printf` to yield strings to the standard output (terminal). Seems like you were not using a good IDE and cannot just find and replace all instances of `printf` with `pprint`.

Task: Given the file test.cpp, you need to replace all instances of `printf` to `pprint`. You need to ensure that ONLY those instances of `printf` are replaced where it is used as a

function.

Usage: `sed -f var_change.sed <filename> > output.txt`

Example:

Input:
```
printf("eren");
otherprintf("mikasa");
```

Output:
```
pprint("eren");
otherprintf("mikasa"); // printf in substring is not changed
```

Note: You need to take filename as argument to the `sed` command.

Problem 3: Things are getting `awk`-ward (did you see what we did there? Oh, okay nevermind.)

You have data of some students in university and being a database administrator you are required to add a column to the given `students.csv` file for generating new email ids of the students.

Task: You are given a file `students.csv` and you have to append a column named `Email-ID` to the database. The entries in email id are formed as `<firstname><lastname>@surveycorps.com`.

Usage: `awk -f email.awk students.csv > output.txt`

Example:

Input:
```
Student ID,First Name,Middle Name,Last Name
1001,Aarav,Rohit,Patel
1002,Ishani,Rani,Sharma
```

Output:
```
Student ID,First Name,Middle Name,Last Name,Email-ID
1001,Aarav,Rohit,Patel,AaravPatel@surveycorps.com
1002,Ishani,Rani,Sharma,IshaniSharma@surveycorps.com
```

Warning: Take care of spaces when using default `awk` configuration.

Problem 4: Another `awk`-ward day.

Data aggregation is an important task on databases in order to represent data using a few numbers (total, average, etc.). `awk` can be used for such aggregating techniques especially in structured data like csv files.

Task: Given the data in `employment.csv` you have to find sum of all entries in **Values** and append it to the end of the file (`Net : <sum>`) and sum for each distinct entry in `High_industry` in the format (`entry_name : <sum>`). Separate the above from the original records using 5 =. See the below example (after printing `Net : <value>`, all other entries should be in lexicographical order).

Usage: `awk -f sum.awk employment.csv > output.txt`

Example:

Input:

```
Week_end,Indicator,High_industry,Value
2019-05-05,Number of paid jobs - 34 days,Film,2090110
2019-05-05,Number of paid jobs - 34 days,Agriculture,95150
2019-05-05,Number of paid jobs - 34 days,Banking,405050
```

Output:

```
Week_end,Indicator,High_industry,Value
2019-05-05,Number of paid jobs - 34 days,Film,2090110
2019-05-05,Number of paid jobs - 34 days,Agriculture,95150
2019-05-05,Number of paid jobs - 34 days,Banking,405050

Net :  354478390
Agriculture :  7888120
Banking :  36859440
Film :  177256480
Manufacturing :  130092910
Railways :  2381440
```

Problem 5: What if its a `sed` life and an `awk`-ward day ...

Transforming Data Styles: With the power of `sed` and `awk`, reshape lines from one format to another effortlessly. Whether it's `sed`'s concise elegance or `awk`'s versatile scripting, these tools bring your data into a new light, converting lines that fit a specific pattern while leaving others untouched. Dive into the world of text manipulation and witness the magic as `sed` and `awk` seamlessly redefine your data's structure.

Task: Given the data in `english.txt`, you have to write an `awk` script `modify.awk` and also a `sed` script `modify.sed` which changes every line which appear in format 1 to format 2. Any line which does not appear in format 1 should be kept as it is.

Format - 1: `<name> got <number> medals in <sport> in <year>`

Format - 2: `In <year>, <name> got <number> medals in <sport>`

Here is the domain of the placeholders:

1. `<name>` : An arbitrary length (> 0) string which contains only alphabets $[A - Za - z]$, no spaces
2. `<number>` : An arbitrary length positive number which contains only digits $[0 - 9]$
3. `<sport>` : An arbitrary length (> 0) string which contains only alphabets $[A - Za - z]$, no spaces
4. `<year>` : A 4-digit number which contains only digits $[0 - 9]$

Note: For simplicity, we are using 1 medals instead of 1 medal so as to reduce the complexity of the task.

Clarification: Both `sed` and `awk` scripts are expected to produce the same output. The input file will contain only lines which are in format 1 or are not in format 1. The output file should contain the lines which are in format 1 in format 2 and the lines which are not in format 1 as it is.

Usage:

```
sed -f modify.sed english.txt > output.txt
awk -f modify.awk english.txt > output.txt
```

Example:

Input:

```
John got 5 medals in Football in 2019
Alice got 1 medals in Swimming in 2018
I am great
Bob got 2 medals in Tennis in 2017
```

Output:

```
In 2019, John got 5 medals in Footballv In 2018, Alice got 1 medals in
Swimming
I am great
In 2017, Bob got 2 medals in Tennis
```