Software Systems Lab: OutLab
LaTeX

Name: Shubham Hazra
Roll no: 210100143

September 4, 2022

Lab 3 - SED and AWK

1 Introduction

SED

Sed is a stream editor. A stream editor is used to perform basic text transformations on an input stream (a file or input from a pipeline). While in some ways similar to an editor which permits scripted edits (such as ed), sed works by making only one pass over the input(s), and is consequently more efficient. But it is sed's ability to filter text in a pipeline which particularly distinguishes it from other types of editors.
The basic syntax is:
sed 'sed_command' filename
For example:
To substitute a certain pattern1 with another pattern2 in a file:
sed 's/pattern1/pattern2/' filename
To print k lines of a file:
sed -n 'kp' filename
To delete k lines of a file:
sed 'kd' filename
For inplace substitutions:
sed -i 's/pattern1/pattern2/' filename

AWK

AWK (awk) is a domain-specific language designed for text processing and typically used as a data extraction and reporting tool. Like sed and grep, it is a filter, and is a standard feature of most Unix-like operating systems. The AWK language is a data-driven scripting language consisting of a set of actions to be taken against streams of textual data – either run directly on files or used as part of a pipeline – for purposes of extracting or transforming text, such as producing formatted reports. The language extensively uses the string datatype, associative arrays (that is, arrays indexed by key strings), and regular expressions. While AWK has a limited intended application domain and was especially designed to support one-liner programs, the language is Turing- complete.
The basic syntax is:
awk 'BEGIN{} {} END{}' filename
For example:
To print contents of a file:
awk '{print}' filename

SED vs AWK

SED	AWK
A command line utility that parses and transforms text, using a simple,compact programming language	A command line utility designed for text processing that allows writing effective programs in the form of statements
Less powerful	More powerful and robust
Simple and limited	Complex and versatile

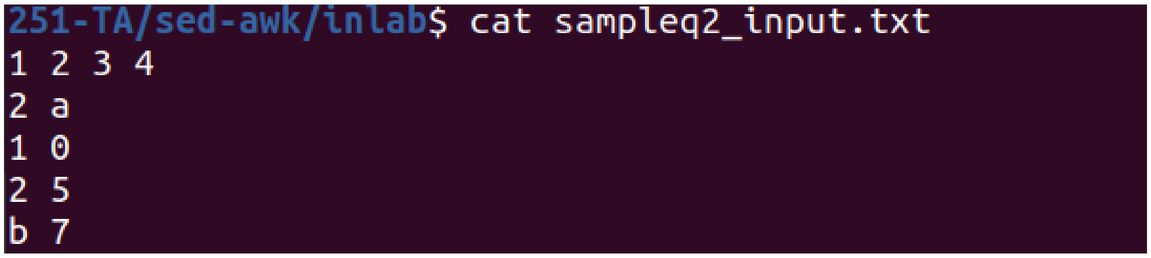
In this lab I was required to write sed commands and awk scripts on bash for the various inlab and outlab problems. There were 3 inlab problems and 5 outlab problems. However I am only going to cover one problem from the inlab and one from the outlab in this report.
The following are the methods I used to solve the problems:

- InLab
    1. Q1_a :- awk
    2. Q1_b :- awk
    3. Q2 :- awk
    4. Q3 :- sed
- OutLab
    1. Q1 :- sed
    2. Q2 :- sed
    3. Q3 :- awk
    4. Q4 :- awk
    5. Q5 :- awk and sed

2 Inlab 3 - Q2

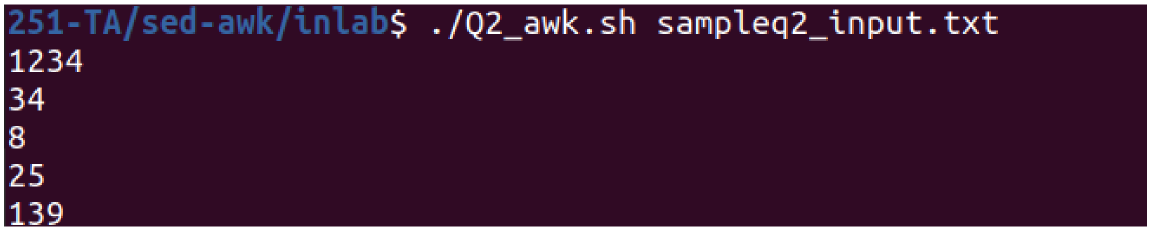
2.1 Problem Statement

Create a function which takes in input a set of numbers from a text file in base 8+2*(line_number%3) and returns the same number in base 10. Note that the digits will be space separated on each line. For digits greater than 9, lowercase alphabets will be used.



```
251-TA/sed-awk/inlab$ cat sampleq2_input.txt
1 2 3 4
2 a
1 0
2 5
b 7
```

Sample Input



```
251-TA/sed-awk/inlab$ ./Q2_awk.sh sampleq2_input.txt
1234
34
8
25
139
```

Sample Output

2.2 Algorithm

Algorithm 1: InLab - Q2

```
$i \leftarrow 10$;
if $i \geq 5$ then
    $i \leftarrow i - 1$;
else
    if $i \leq 3$ then
        $i \leftarrow i + 2$;
    end
end
```

2.3 Code

InLab - Q2

```
#!/bin/bash

awk '
{
    num_digits=NF;
    for(i=1;i<=NF;i++)
    {
        if($i=="a")
        {digits[i-1]=10}
        else if($i=="b")
                {digits[i-1]=11}
        else if($i=="c")
                {digits[i-1]=12}
        else {digits[i-1]=$i}
    }
    base=8+2*(NR%3)
    sum=0;
    for(i=0;i<num_digits;i++)
    {
        mult=1;
        for(j=0; j<num_digits-i-1; j++)
        {
            mult=mult*base;
        }
        sum=sum+digits[i]*mult;
    }
    print sum
}' $1
```

2.4 Results

3 Outlab 3 - Q4

3.1 Problem Statement

Your task for this question is to perform addition of two numbers and write the result to file for each test case.

Sample Input:

```
1
10 2
9 9 9
1 0 0
```

Input description:
First line of the file indicates the number of test cases. Second Line gives the base of the input and output separated by spaces. The next two lines of each test case have two numbers. The digits of the numbers are space separated. The base is guaranteed to be less than or equal 16.

Expected output for the above input:

```
1 0 0 0 1 0 0 1 0 1 1
```

Output description:
The digits in each result should be space separated. Each line should contain only one result.

Write the bash script q4.sh that takes two text files, one containing input and other to write the output. Make use of **awk** (mandatory!) to solve the task.

3.2 Algorithm

Algorithm 2: OutLab - Q4

```
$i \leftarrow 10$;
**if** $i \geq 5$ **then**
    $i \leftarrow i - 1$;
**else**
    **if** $i \leq 3$ **then**
        $i \leftarrow i + 2$;
    **end**
**end**
```

3.3 Code

OutLab - Q4

```
#!/bin/bash

awk '{
    if(NR==1)
    {
        testcases=$1
    }
    if(NR%3==2)
    {
        input_base=$1;
        output_base=$2;
    }
    if(NR%3==0)
    {
        num_digits1=NF;
        for(i=1;i<=NF;i++)
        {
            if($i>=0 && $i<=9)
            {digits1[i-1]=$i}
            else if($i=="a")
            {digits1[i-1]=10}
            else if($i=="b")
            {digits1[i-1]=11}
            else if($i=="c")
            {digits1[i-1]=12}
            else if($i=="d")
            {digits1[i-1]=13}
            else if($i=="e")
            {digits1[i-1]=14}
            else if($i=="f")
            {digits1[i-1]=15}
            else
            {digits1[i-1]=$i}
        }
        num1=0;
```

```
            for(i=0;i<num_digits1;i++)
            {
                mult=1;
                for(j=0; j<num_digits1-i-1; j++){
                mult=mult*input_base;
                }
                num1=num1+digits1[i]*mult;
            }
            }
            if(NR%3==1 && NR!=1)
            {
                num_digits2=NF;
                for(i=1;i<=NF;i++)
                {
                    if($i>=0 && $i<=9)
                    {digits2[i-1]=$i}
                    else if($i=="a")
                    {digits2[i-1]=10}
                    else if($i=="b")
                    {digits2[i-1]=11}
                    else if($i=="c")
                    {digits2[i-1]=12}
                    else if($i=="d")
                    {digits2[i-1]=13}
                    else if($i=="e")
                    {digits2[i-1]=14}
                    else if($i=="f")
                    {digits2[i-1]=15}
                    else
                    {digits2[i-1]=$i}
                }
                num2=0;
                for(i=0;i<num_digits2;i++)
                {
                    mult=1;
                    for(j=0; j<num_digits2-i-1; j++){
                    mult=mult*input_base;
                    }
                    num2=num2+digits2[i]*mult;
                }
                sum = num1+num2;
                num_digits=0
                i=0
                while(sum>0)
                {
                    digits[i]=int(sum%output_base)
                    sum=int(sum/output_base)
                    num_digits++
                    i++
                }
                for(i=num_digits-1;i>=0;i--)
                {
                    printf "%d ",digits[i]

                }
                printf "\n"
            }
    }' $1 > $2
```

3.4 Results