Operating System
LAB 01

4.
(Answer)

First Command:
sed 's/^.\|.$//g' hello (This will remove first and last character of each line of hello file)
or less hello | sed 's/^.\|.$//g'
or cat hello | sed 's/^.\|.$//g'
or more hello | sed 's/^.\|.$//g'

Second Command:

grep -iwc ‘hello’ hellofile
or less hellofile | grep -ic ‘hello’
or more hellofile | grep -iwc ‘hello’
or cat hellofile | grep -ic ‘hello’

which finds how many lines of a file contain a Given word.

man sed

sed - stream editor for filtering and transforming text.

sed options **scripts** filename

sed ‘s/hello/world/’ hello > output
or sed ‘s/hello/world/’ < hello > output
or cat hello | sed ‘s/hello/world/’ - > output

to replace all occurrences of 'hello' to 'world' in the hello file.

sed -i 's/hello/world/' hello
-i to edit files in-place instead of printing to standard output.

sed 's/hello/world/' hello
This command will replace 'world' with 'hello' and just print it. But **it will not change text in the hello file.**

sed -n '8p' hello
-n to suppress output and **p** command to print specific lines.

sed -n '1p ; $p' hello output
or sed -n '1p;$p' hello output output1
Print first line of hello **1p** and **$p** means last line of last file.

echo 's/hello/world/g' > myscript.sed
or sed -f myscript.sed input.txt > output.txt
or sed --file=myscript.sed input.txt > output.txt
Data will store into output.txt.

Or sed -f myscript.sed input.txt
This command will only print but not change in input.txt.

sed '1,4d' input.txt
This command will delete first to forth line of the input.txt.

sed '2,$ s/hi/hello/g' hellofile
only second and last line will be replaced.
$ means last line.

sed '/^hel/q34' hello

This command print **hello** until ‘hel’ is found.

sed ‘/^hel/d ; s/hello/world/g’ hello
or
sed -e ‘/^hel/d’ -e ‘s/hello/world/g’ hello
or
echo ‘s/hello/world/g’ > myscript.sed
sed -e ‘/^hel/d’ -f myscript.sed hello
or
echo ‘/^hel/d ; s/hello/world/g’ > myscript.sed
sed -f myscript.sed hello
All ‘hel’ word including line will be deleted and ‘hello’ will be replaced by ‘world’ and with the **‘g’** word, it will be replaced globally.
Hello file will be printed.

seq 3 | sed 2q or more hello | sed 2q
seq means sequence and this will print 1 to 3 number. But sed will terminate when it will reach to 2 / second line.

seq 3 | sed 2d or less hello | sed 2d
It will delete second sequence / second line .

more hello | sed -n 2p
This command will print only second line of hello file.

less hello | sed ‘n;n;s/./kill/’
Every third line of hello, perform substitution the first letter with ‘kill’.

seq 6 | sed ‘n;n;s/./kill/’
or seq 6 | sed ‘0~3s/./kill/’ (0
1
2
kill

4
5
kill

seq | sed ‘1~2s/./kill/’ (0th and 2th will suffer)

seq | sed ‘1~3s/./kill/’ (0th and 3th will suffer)

seq 3 | sed -n '2{s/2/X/ ; p}’
This will only print second with a substitution of X.
X

more linux | sed ‘2a hello’
or sed ‘2a hello’ linux
This command will add hello after the second line.
‘a TEXT’ means appending TEXT after a line.

```
# more linux | sed ‘2a\
        hello\
        world
        3s/./hey/’
```
This command will print hello after second line and then print world and then substitute 3 with hey.
unix is great os.
linux is opensource.
hello
world
heynix is a free os.

sed ‘2i hello’ hellofile
This command will include ‘hello’ before second line.

```
# sed ‘2i\
        hello
        world\
        3s/./hey/’ hellofile
```
‘hello’ and ‘world’ will be printed before second line. Third line first letter will be replaced by ‘hey’
unix is great os.
hello
world
linux is opensource.
heynix is free os.

i append the character before the line.

```
# sed ‘2,9c hello’ hellofile
```
2th to 9th lines will be replaced by ‘hello’
unix is great os.
Hello

c TEXT replaces stuffs.

```
# sed ‘2,9c\ delete’ hellofile
```
This is hellofile.
Delete

c deletes the matching addresses.

```
# seq 3 | sed '2c\
       hello
        s/./X/'
```
X
hello
X

```
# sed -e '$s/./two/' -e '1{s/./one/ ; }' hellofile
```
This command will replace last line first character and first line first character.

onenix is great os.
linux is opensource.
unix is free os.
learn operating system.
unix linux which one you choose.
unix is easy to learn.
linux is a multiuser os. Learn unix .unix is a powerful.
twooo is life.

sed G linux
Insert one blank line after each line.

sed ‘G;G’ linux
two blank lines.

sed '/unix/{x;p;x;}' linux
Insert a black line above every line which matches “linux”

unix is great os.
linux is opensource.

unix is free os.
learn operating system.

unix linux which one you choose.
foo is life.

unix is easy to learn.

linux is a multiuser os. Learn unix .unix is a powerful.
foo is life.

sed ‘/unix/G’ linux
Insert a blank after every line.

sed '/life/d' linux
‘life’ matching line will be deleted.

unix is great os.
linux is opensource.
unix is free os.
learn operating system.
unix linux which one you choose.
unix is easy to learn.
linux is a multiuser os. Learn unix .unix is a powerful.

sed ‘s/[Uu]inx/Linux/g’ hello
All uppercase/ lowercase will be replaced.

```
# seq 6 | sed -e 1d -e 3d -e 5d
    2
    4
    6

# seq 6 | sed '1d;3d;5d'
    2
    4
    6

# seq 6 | sed ‘{1d;3d;5d}’
2
4
6

# seq 2 | sed '1a\
        Hello
        2d'
```

prints
1
Hello

ls -log or ls -l works as same.

seq 2| sed ‘1w hello.txt ; 2d ‘
1
2
This command will not work properly due to ‘;’

sed '/apple/s/hello/world/' input.txt > output.txt
This command replaces the word 'hello' with 'world' only in lines containing the word 'apple'.

First ~ Step
1 ~ 2 means first =1 and step is 2 . That means every odd number line will be counted. (1,3,5,7,9)

2~3 means to pick every third line starting with the second. (2,5,8)

10 ~ 5 means to pick every fifth line starting with the tenth.(10,15,20,25)

0 ~ 4 means even number will be counted with 4 steps.

seq 10 | sed -n '0~4p'
4
8

seq 10 | sed ‘1~3p’
1
4
7
10

sed -n '\;^/home/alice/documents/;p'
They print lines which start with '/home/alice/documents/'.

sed -n '/hello/p' hellofile
This command prints lines containing the word 'hello'.
grep 'hello' hellofile

@@
The syntax of the s command (which stands for "substitute") is:
'**s/regexp/replacement/flags**'.

Character	Does
*	Matches a sequence of zero or more instances of matches for the preceding regular expression.
\{i,j\}	Matches between i and j, inclusive, sequences.
.	Matches any character, including a newline.
^	Matches the null string at beginning of the pattern space.
$	It refers to end of pattern space.
regexp1|regexp2	Matches either regexp1 or regexp2. Left to right.

sed 's/^.\|.$//g' hello

Matches either **^.** (first) or **.$** (last) can be written as **^.\|.$**
s/**^.\|.$**/(null)/(globally)

This command will remove first and last character of each line of hello file.

The man who want to know the next command should be, I prefer them to get these stuffs on PhD course on Linux shits.

man grep

grep, which stands for "global regular expression print," processes text line by line and prints any lines which match a specified pattern.

grep options pattern filename

less linux | grep ‘unix’
or grep ‘unix’ linux

This command will print all ‘unix’ word in color of the linux file.

grep -n ‘unix’ linux
-n will show the line number where the pattern will be matched.

grep -ni ‘unix’ linux or
grep --color -ni ‘unix’ *.txt
-i will show all stuffs in case-insensitive.
***.txt** will find the pattern all the txt file exists in current directory.

grep -nir ‘unix’ or grep -nir ‘unix’ Documents
-r will recursively find all the pattern in the current directory.

grep -nr -e ‘eesha’ -e ‘esha’ -e ‘EeSha’ Documents
or
cat > pattern
eesha
esha
EeSha
Then,
grep -nr -f pattern Documents

grep -inrc ‘unix’
Count the pattern with line number and file name, recursively.
Prints like:
statistics:0

vim:0
hello:0
output:0
myscript.sed:1

Some options:

-c : This prints only a count of the lines that match a pattern
-h : Display the matched lines, but do not display the filenames.
-i : Ignores, case for matching
-l : Displays list of a filenames only.
-n : Display the matched lines and their line numbers.
-v : This prints out all the lines that do not matches the pattern
-f file : Takes patterns from file, one per line.
-w : Match whole word.
-r : recursively print all stuffs.

man awk

awk stands for "Aho, Weinberger, and Kernighan".
awk is an interpreted programming language which focuses on processing text.

awk [-F fs] [-v var=value] ['prog' | -f progfile] [file ...]

-F fs	Sets the input field separator to the regular expression fs.
-v var=value	Assigns the value value to the variable var before executing the awk program.
'prog'	An awk program itself.
-f progfile	Awk program file.
File….	A file.

awk ‘length($0) >72’ hellofile
Print only lines that are longer than 72 characters.

length() is a function.
An empty expression-list stands for **$0**.
‘length($0) >72’ is a ‘awk program’.

awk ‘{print $2, $1}’ hellofile
This command will swap first word of a line.

awk -f prog.awk filename
where prog.awk is

```
{ s += $1 }
END { print "sum is", s, " average is", s/NR }
```

This program will add up first column of its input file, and print the sum and average of the values.

Or

```
BEGIN {
for (i = 1; i < ARGC; i++) printf "%s ", ARGV[i]
printf "\n"
exit }
```

This awk program simulates the echo command.

END