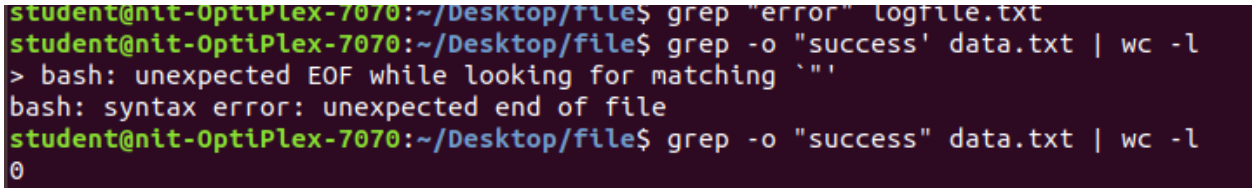
UNIX AND SHELL SCRIPTING LAB 4

ROLL NO:423111

1.Using `grep`

1. Find all lines containing the word **"error"** in a log file (`log.txt`).
2. Count the occurrences of the word **"success"** in a file (`data.txt`).
3. Extract all lines from a file (`records.txt`) that start with a digit.
4. Display all lines in `file.txt` that do **not** contain the word **"failed"**.
5. Find all `.txt` files in the current directory that contain the word **"TODO"**.
6. Extract email addresses from `contacts.txt` (Hint: Use regex).
7. Find all occurrences of **"apple"**, case-insensitive, in `fruits.txt`.
8. Find all lines in `logfile.txt` that contain **either "error" or "fail"**.
9. Display lines that **start with a capital letter** from `sentences.txt`.
10. List only filenames from the current directory that contain the word **"project"**.(you can pick any word here that is being repeated)
11. Find lines in `server.log` that contain **"404"**, but ignore case sensitivity.
12. Find all words in `dictionary.txt` that **end with "ing"**.
13. Extract **dates (YYYY-MM-DD format)** from `events.txt`.



```
student@nit-OptiPlex-7070:~/Desktop/file$ grep -E "^[0-9]" records.txt
101,John Doe,Manager,50000
102,Alice Smith,Developer,60000
103,Bob Brown,Designer,55000
104,Charlie Johnson,Analyst,52000
105,David White,Developer,62000
106,Eve Black,Manager,70000
student@nit-OptiPlex-7070:~/Desktop/file$ grep -v "failed" file.txt
The quick brown fox jumps over the lazy dog.
A journey of a thousand miles begins with a single step.
Hello world! This is a simple test file.
Sed and awk are powerful text-processing tools.
Regular expressions are very useful in scripting.
This file contains multiple lines for testing purposes.


student@nit-OptiPlex-7070:~/Desktop/file$ grep -l "TODO" *.txt
student@nit-OptiPlex-7070:~/Desktop/file$ grep -E -o "[a-zA-z0-9._%+-]+@[a-zA-Z0-9.-]+\.
> bash: unexpected EOF while looking for matching `"'
bash: syntax error: unexpected end of file
student@nit-OptiPlex-7070:~/Desktop/file$ grep -E -o "[a-zA-z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}" contacts.txt
grep: Invalid range end
student@nit-OptiPlex-7070:~/Desktop/file$ grep -E -o "[a-zA-z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2, }" contacts.txt
grep: Invalid range end
student@nit-OptiPlex-7070:~/Desktop/file$ grep -i "apple" fruits.txt
apple
student@nit-OptiPlex-7070:~/Desktop/file$ grep -E "error|fail" logfile.txt
[2024-02-01 12:15:50] ERROR: User authentication failed.
student@nit-OptiPlex-7070:~/Desktop/file$ grep -E "^[A-Z]" sentences.txt
The quick brown fox jumps over the lazy dog.
A journey of a thousand miles begins with a single step.
Hello world! This is a simple test file.
Sed and awk are powerful text-processing tools.
Regular expressions are very useful in scripting.
This file contains multiple lines for testing purposes.
student@nit-OptiPlex-7070:~/Desktop/file$ grep -l "project" *
student@nit-OptiPlex-7070:~/Desktop/file$ grep -i "404" server.log
192.168.1.13 - - [10/Feb/2024:10:19:21] "GET /contact.html HTTP/1.1" 404
student@nit-OptiPlex-7070:~/Desktop/file$ grep -E "\bing\b" dictionary.txt
student@nit-OptiPlex-7070:~/Desktop/file$ grep -E -o "[0-9]{4}-[0-9]{2}-[0-9]{2}" events.txt
2024-01-01
2024-02-14
2024-07-04
2024-12-25
```

```
student@nit-OptiPlex-7070:~/Desktop/file$ grep -E -o "[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}" contacts.txt
john.doe@gmail.com
alice.smith@yahoo.com
bob.brown@outlook.com
charlie.johnson@gmail.com
david.white@hotmail.com
eve.black@company.com
frank.green@university.edu
```

2. Using `sed`

1. Replace all occurrences of **"foo"** with **"bar"** in `text.txt`.
2. Delete all blank lines from `input.txt`.
3. Remove leading and trailing spaces from each line in `whitespace.txt`.
4. Insert a new line with the text **"Header: Report"** at the beginning of `report.txt`.
5. Replace all instances of multiple spaces with a single space in `file.txt`.
6. Swap the first and second word in each line of `swap.txt`.
7. Remove everything after the first comma in each line of `csv_data.txt`.
8. Replace the word **"old"** with **"new"**, but only on lines that contain the word **"update"**.
9. Delete all occurrences of a number from `text.txt`.
10. Convert all lowercase letters to uppercase in `names.txt`.
11. Replace all **dates in DD-MM-YYYY format** with **YYYY-MM-DD** in `dates.txt`.
12. Add **line numbers** at the beginning of each line in `story.txt`.
13. Surround all words in `title.txt` with double quotes (").

```
student@nit-OptiPlex-7070:~/Desktop/file$ sed 's/foo/bar/g' text.txt
Hello world! This is a simple text file.
It contains multiple lines.
Some words are repeated, repeated multiple times.
This is a great way to test text processing.


student@nit-OptiPlex-7070:~/Desktop/file$ sed '/^$/d' input.txt
Alice 25 Engineer 60000
Bob 30 Doctor 80000
Charlie 28 Teacher 50000
David 35 Lawyer 90000
Eve 27 Scientist 75000
Frank 40 Pilot 100000
student@nit-OptiPlex-7070:~/Desktop/file$ sed 's/^[ \t]*//;s/[ \t]*$//' whitespace.txt
Alice     25     Engineer     60000
Bob  30 Doctor  80000
Charlie      28 Teacher     50000
David  35 Lawyer     90000
Eve   27  Scientist   75000
Frank   40    Pilot   100000


student@nit-OptiPlex-7070:~/Desktop/file$ sed '1i\Header: Report' report.txt
Header: Report
Eve         North      20000
Frank       South      19000
Sales Report - January 2025

Employee    Region     Sales
Alice       North      15000
Bob         South      18000
Charlie     East       17000
David       West       16000

student@nit-OptiPlex-7070:~/Desktop/file$ sed 's/ */ /g' file.txt
 T h e q u i c k b r o w n f o x j u m p s o v e r t h e l a z y d o g .
 A j o u r n e y o f a t h o u s a n d m i l e s b e g i n s w i t h a s i n g l e s t e p .
 H e l l o w o r l d ! T h i s i s a s i m p l e t e s t f i l e .
 S e d a n d a w k a r e p o w e r f u l t e x t - p r o c e s s i n g t o o l s .
 R e g u l a r e x p r e s s i o n s a r e v e r y u s e f u l i n s c r i p t i n g .
 T h i s f i l e c o n t a i n s m u l t i p l e l i n e s f o r t e s t i n g p u r p o s e s .
```

```
student@nit-OptiPlex-7070:~/Desktop/file$ sed 's/•[^ ]*• •[^ ]*•/\2 \1/' swap.txt
sed: -e expression #1, char 32: invalid reference \2 on `s' command's RHS
student@nit-OptiPlex-7070:~/Desktop/file$ sed 's/,.*//' csv_data.txt
ID
101
102
103
104
105
106


student@nit-OptiPlex-7070:~/Desktop/file$ sed '/update/s/old/new/g' text.txt
Hello world! This is a simple text file.
It contains multiple lines.
Some words are repeated, repeated multiple times.
This is a great way to test text processing.


student@nit-OptiPlex-7070:~/Desktop/file$ sed 's/[0-9]//g' text.txt
Hello world! This is a simple text file.
It contains multiple lines.
Some words are repeated, repeated multiple times.
This is a great way to test text processing.


student@nit-OptiPlex-7070:~/Desktop/file$
student@nit-OptiPlex-7070:~/Desktop/file$ sed 's/[0-9]//g' text.txt
Hello world! This is a simple text file.
It contains multiple lines.
Some words are repeated, repeated multiple times.
This is a great way to test text processing.
```

3.**Using** `awk`

1. Print only the second column from a space-separated file (`data.txt`).
2. Sum the numbers in the third column of `values.txt`.
3. Count the number of lines in `log.txt` that contain the word **"warning"**.
4. Print all lines in `marks.txt` where the second column is greater than 50.
5. Print only the first and last columns from a tab-separated file (`data.csv`).
6. Calculate and print the average of the numbers in the second column of `numbers.csv`.
7. Print all lines in `students.csv` where the **third column (marks)** is greater than **75**.
8. Print the sum of all numbers in the first column of `data.txt`.
9. Display the last column of `students.csv`, where columns are separated by commas.
10. Print lines where the second column starts with the letter "A".
11. Find the **highest number** in the third column of `stats.txt`.
12. Count how many lines contain a word longer than 10 characters in `words.txt`.
13. Extract domain names from an email list (`emails.txt`).

```
student@nit-OptiPlex-7070:~/Desktop/file$ awk '{print $2}' data.txt
50
50
45
70
55


student@nit-OptiPlex-7070:~/Desktop/file$ awk '{sum += $3} END {print sum}' values.txt
300
student@nit-OptiPlex-7070:~/Desktop/file$ awk '/warning/ {count++} END {print count}' log.txt
awk: cannot open log.txt (No such file or directory)
student@nit-OptiPlex-7070:~/Desktop/file$ awk '$2 > 50' marks.txt
101 85
102 75
103 60
104 90
105 78
student@nit-OptiPlex-7070:~/Desktop/file$ awk -F'\t' '{print $1, $NF}' data.csv
```

```
student@nit-OptiPlex-7070:~/Desktop/file$ awk -F',' '$3 > 75' students.csv
awk: cannot open students.csv (No such file or directory)
student@nit-OptiPlex-7070:~/Desktop/file$ awk '{sum += $1} END {print sum}' data.txt
15
student@nit-OptiPlex-7070:~/Desktop/file$ awk -F',' '{print $NF}' students.csv
awk: cannot open students.csv (No such file or directory)
student@nit-OptiPlex-7070:~/Desktop/file$ awk '$2 ~ /^A/' file.txt
student@nit-OptiPlex-7070:~/Desktop/file$ awk '{if ($3 > max) max = $3} END {print max}' stats.txt
awk: cannot open stats.txt (No such file or directory)
student@nit-OptiPlex-7070:~/Desktop/file$ awk '{for (i=1; i<=NF; i++) if (length($i) > 10) {count++; break}} END {print count}' words.txt
1
student@nit-OptiPlex-7070:~/Desktop/file$
student@nit-OptiPlex-7070:~/Desktop/file$ awk -F'[@]' '{print $2}' emails.txt
gmail.com
yahoo.com
outlook.com
gmail.com
hotmail.com
```