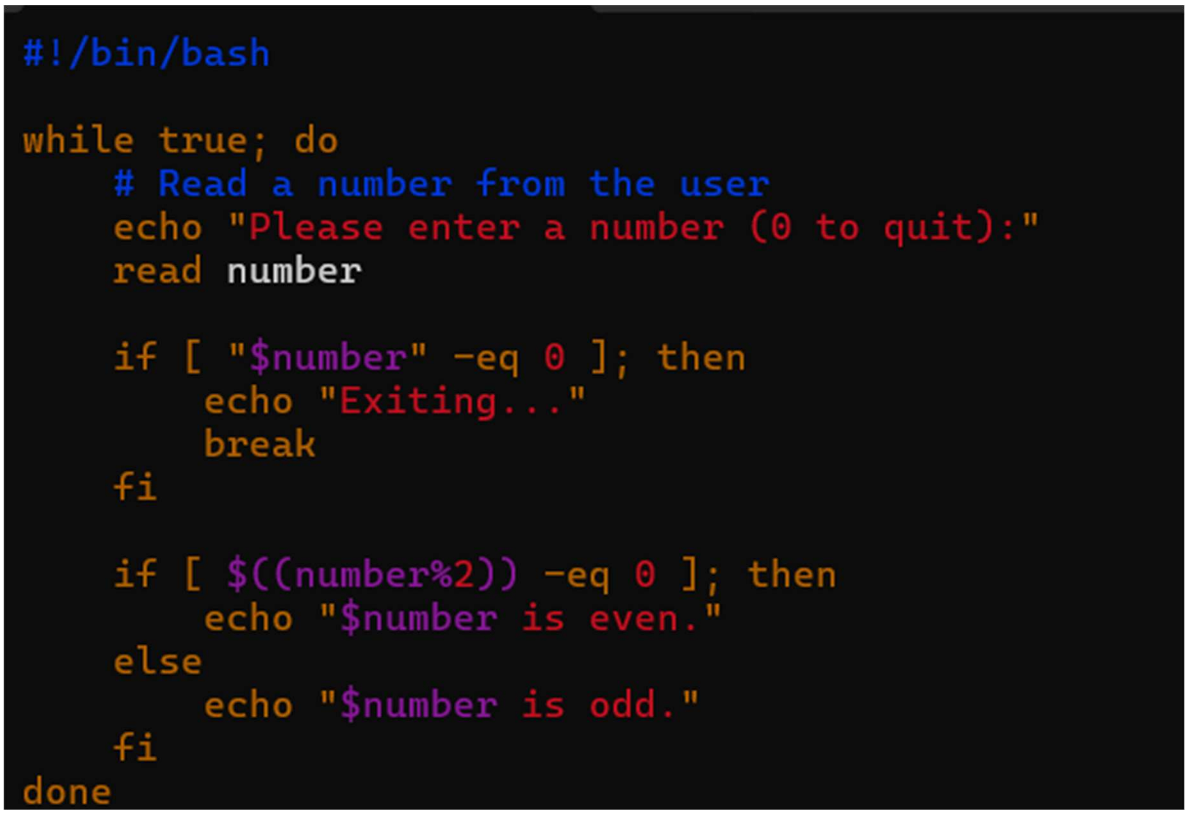
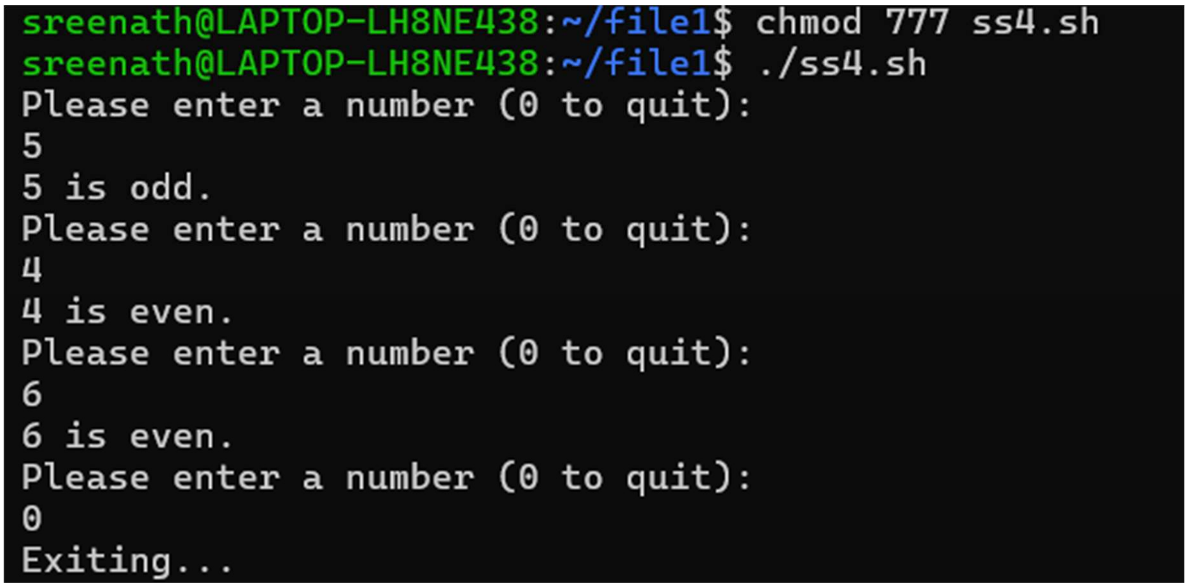
Assignment 1: Ensure the script checks if a specific file (e.g., myfile.txt) exists in the current directory. If it exists, print "File exists", otherwise print "File not found".
Answer:

```
filename="myfile.txt"

#!/bin/bash

if [ -e "$filename" ]; then
    echo "file exists"
else
   echo "file not found."
fi
```

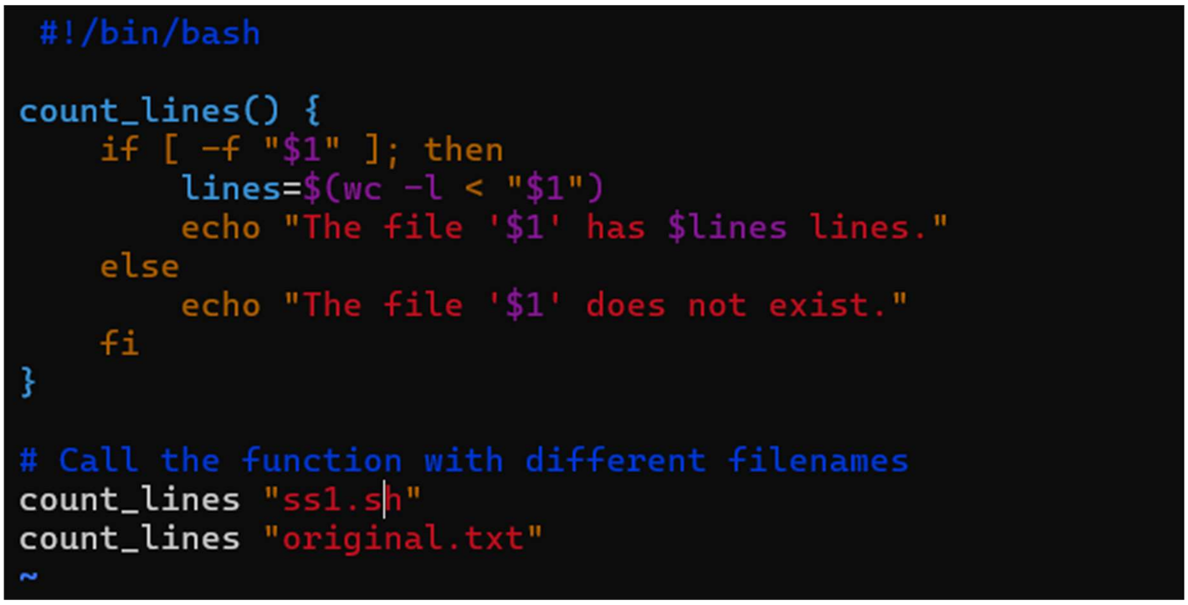
Assignment 2: Write a script that reads numbers from the user until they enter '0'. The script should also print whether each number is odd or even.



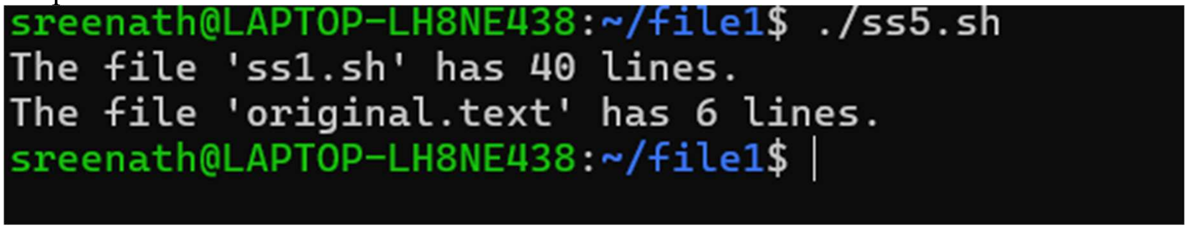
```
#!/bin/bash

while true; do
    # Read a number from the user
    echo "Please enter a number (0 to quit):"
    read number

    if [ "$number" -eq 0 ]; then
        echo "Exiting..."
        break
    fi

    if [ $((number%2)) -eq 0 ]; then
        echo "$number is even."
    else
        echo "$number is odd."
    fi
done
```


```
sreenath@LAPTOP-LH8NE438:~/file1$ chmod 777 ss4.sh
sreenath@LAPTOP-LH8NE438:~/file1$ ./ss4.sh
Please enter a number (0 to quit):
5
5 is odd.
Please enter a number (0 to quit):
4
4 is even.
Please enter a number (0 to quit):
6
6 is even.
Please enter a number (0 to quit):
0
Exiting...
```

Assignment 3: Create a function that takes a filename as an argument and prints the number of lines in the file. Call this function from your script with different filenames.

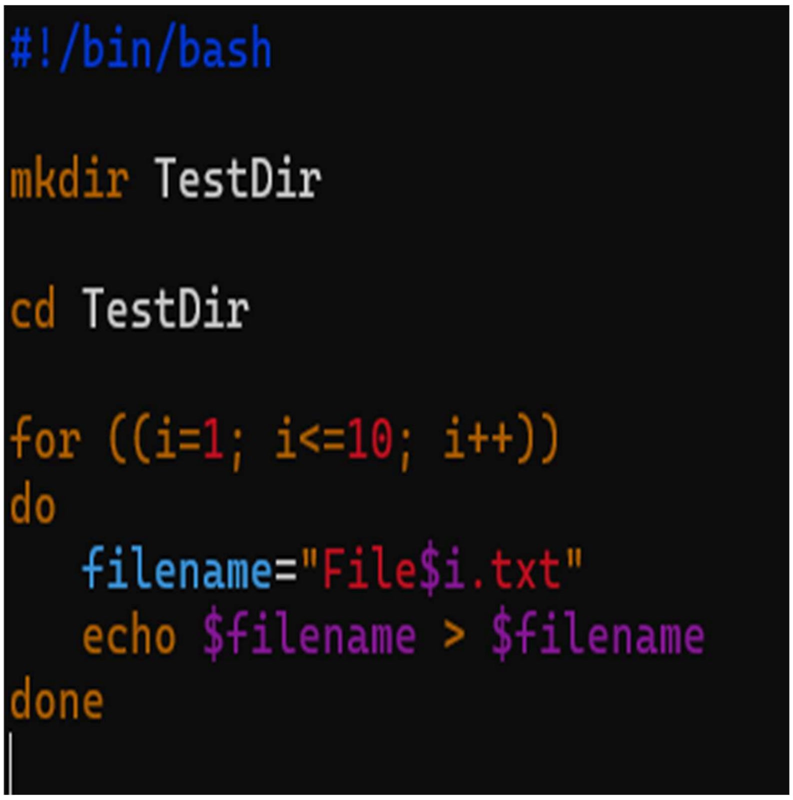

```
 #!/bin/bash

count_lines() {
    if [ -f "$1" ]; then
        lines=$(wc -l < "$1")
        echo "The file '$1' has $lines lines."
    else
        echo "The file '$1' does not exist."
    fi
}

# Call the function with different filenames
count_lines "ss1.sh"
count_lines "original.txt"
~
```

Output:


```
sreenath@LAPTOP-LH8NE438:~/file1$ ./ss5.sh
The file 'ss1.sh' has 40 lines.
The file 'original.text' has 6 lines.
sreenath@LAPTOP-LH8NE438:~/file1$ |
```

This script defines a function count_lines that takes a filename as an argument. It checks if the file exists using the -f test. If the file exists, it uses the wc -l command to count the number of lines in the file and prints the result. If the file does not exist, it prints a message indicating that the file does not exist.

Assignment 4: Write a script that creates a directory named TestDir and inside it, creates ten files named File1.txt, File2.txt, ... File10.txt. Each file should contain its filename as its content (e.g., File1.txt contains "File1.txt").



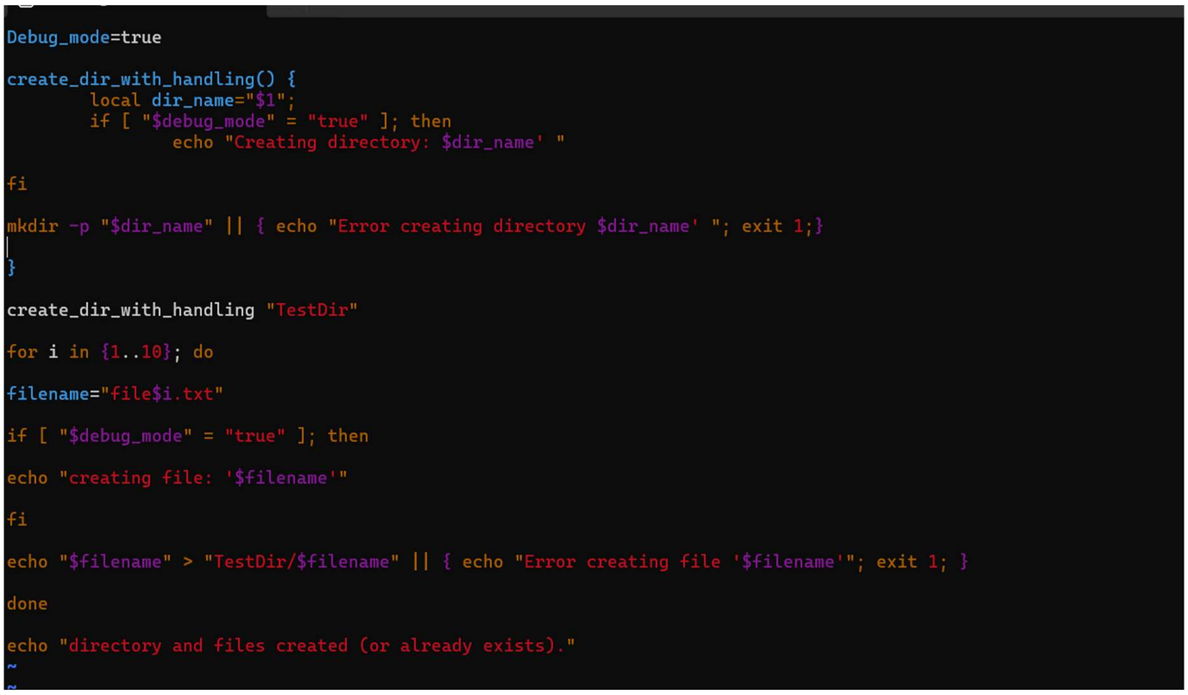
```
#!/bin/bash

mkdir TestDir

cd TestDir

for ((i=1; i<=10; i++))
do
   filename="File$i.txt"
   echo $filename > $filename
done
```

Firstly we want to creates a directory named TestDir using the mkdir command. It then changes to the TestDir directory using the cd command. Finally, it creates ten files named File1.txt, File2.txt, …, File10.txt using a for loop. The echo command is used to write the filename into each file.

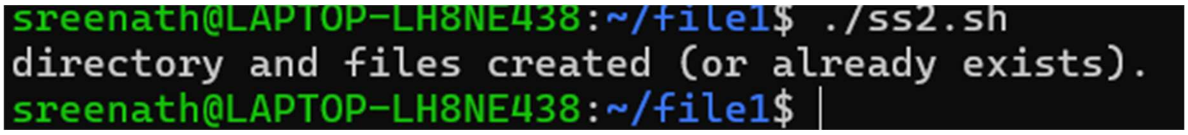
To run the script, you may need to give it execute permissions using the command chmod +x scriptname.sh. Then you can run it with ./scriptname.sh.

Assignment 5: Modify the script to handle errors, such as the directory already existing or lacking permissions to create files.

Add a debugging mode that prints additional information when enabled.

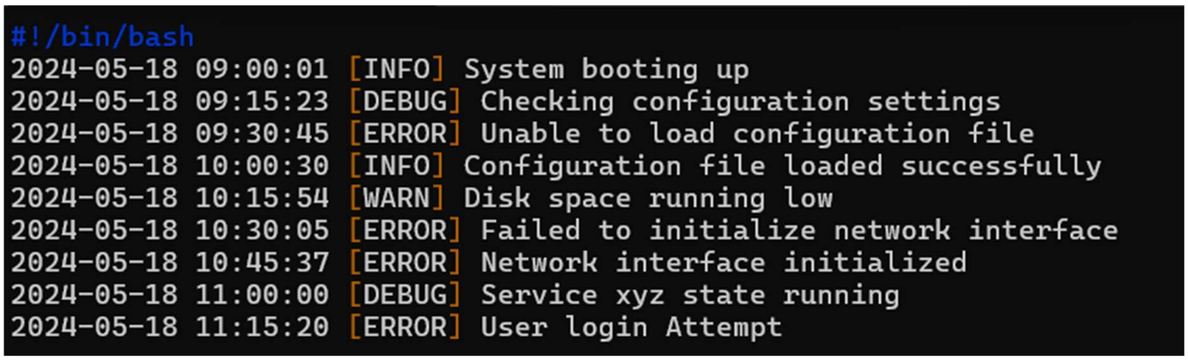


```
Debug_mode=true

create_dir_with_handling() {
        local dir_name="$1";
        if [ "$debug_mode" = "true" ]; then
                echo "Creating directory: $dir_name' "

fi

mkdir -p "$dir_name" || { echo "Error creating directory $dir_name' "; exit 1;}
|
}

create_dir_with_handling "TestDir"

for i in {1..10}; do

filename="file$i.txt"

if [ "$debug_mode" = "true" ]; then

echo "creating file: '$filename'"

fi

echo "$filename" > "TestDir/$filename" || { echo "Error creating file '$filename'"; exit 1; }

done

echo "directory and files created (or already exists)."
~
~
```



```
sreenath@LAPTOP-LH8NE438:~/file1$ ./ss2.sh
directory and files created (or already exists).
sreenath@LAPTOP-LH8NE438:~/file1$ |
```

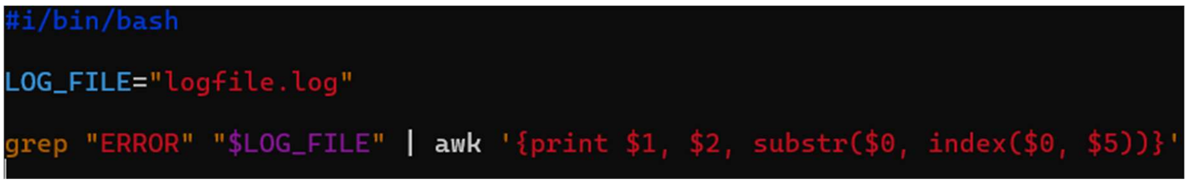
Assignment 6: Given a sample log file, write a script using grep to extract all lines containing "ERROR". Use awk to print the date, time, and error message of each extracted line.
Data Processing with sed
Answer:
Firstly, I have created a sample log file i.e logfile.log and enter some error message with date and time



```
#!/bin/bash
2024-05-18 09:00:01 [INFO] System booting up
2024-05-18 09:15:23 [DEBUG] Checking configuration settings
2024-05-18 09:30:45 [ERROR] Unable to load configuration file
2024-05-18 10:00:30 [INFO] Configuration file loaded successfully
2024-05-18 10:15:54 [WARN] Disk space running low
2024-05-18 10:30:05 [ERROR] Failed to initialize network interface
2024-05-18 10:45:37 [ERROR] Network interface initialized
2024-05-18 11:00:00 [DEBUG] Service xyz state running
2024-05-18 11:15:20 [ERROR] User login Attempt
```

After this create script to extract and format the lines containing “ERROR”.

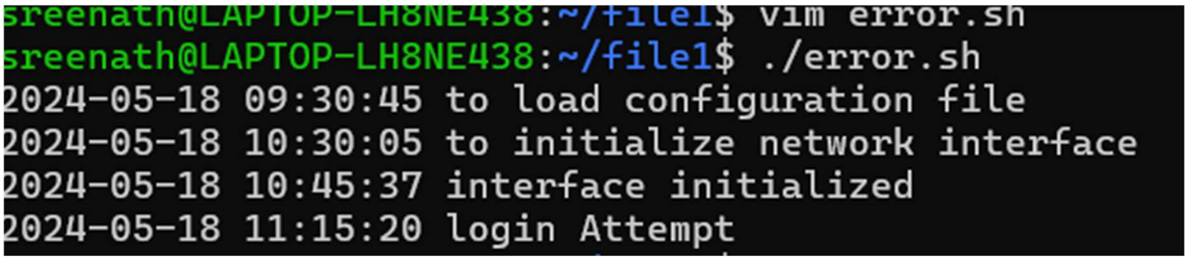
Script name: error.sh



```
#i/bin/bash

LOG_FILE="logfile.log"

grep "ERROR" "$LOG_FILE" | awk '{print $1, $2, substr($0, index($0, $5))}'
```

grep "ERROR" "$LOG_FILE": This extracts all lines containing the word "ERROR". awk '{print $1, $2, substr($0, index($0,$5))}': This processes each extracted line to print the date, time, and the error message.
Specifically: $1: The date part (e.g., 2024-05-18).
$2: The time part (e.g., 10:25:33).
substr($0, index($0,$5)): This extracts the substring starting from the fifth field onward (which includes the error message.)

Output:

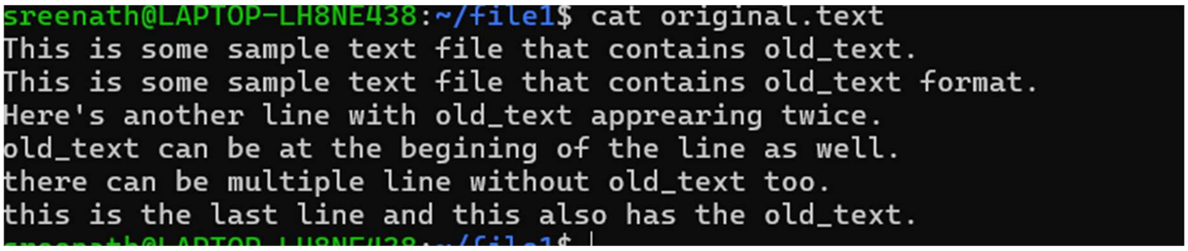


```
sreenath@LAPTOP-LH8NE438:~/file1$ vim error.sh
sreenath@LAPTOP-LH8NE438:~/file1$ ./error.sh
2024-05-18 09:30:45 to load configuration file
2024-05-18 10:30:05 to initialize network interface
2024-05-18 10:45:37 interface initialized
2024-05-18 11:15:20 login Attempt
```

Assignment 7: Create a script that takes a text file and replaces all occurrences of "old_text" with "new_text". Use sed to perform this operation and output the result to a new file.

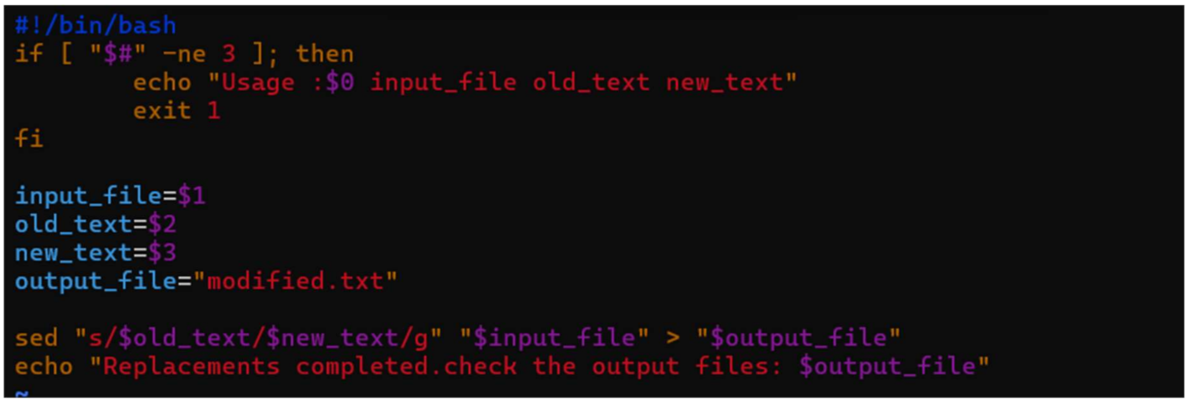
Answer:
We need to create two files, one is the original file for storing the content and other one for modified file.
Then we need to create a replace script for writing the script.

Orginal.text content:



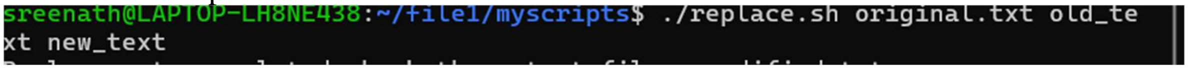
```
sreenath@LAPTOP-LH8NE438:~/file1$ cat original.text
This is some sample text file that contains old_text.
This is some sample text file that contains old_text format.
Here's another line with old_text apprearing twice.
old_text can be at the begining of the line as well.
there can be multiple line without old_text too.
this is the last line and this also has the old_text.
```

Script:



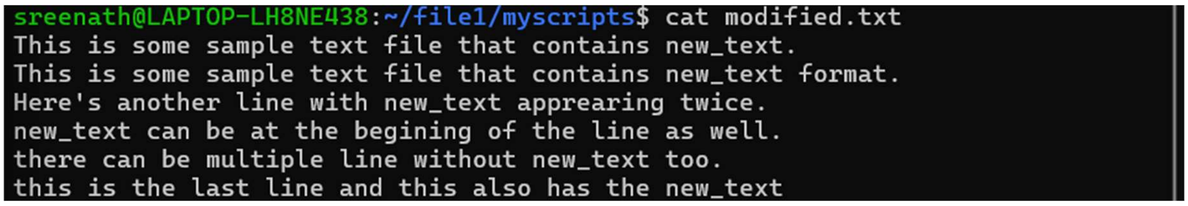
```
#!/bin/bash
if [ "$#" -ne 3 ]; then
        echo "Usage :$0 input_file old_text new_text"
        exit 1
fi

input_file=$1
old_text=$2
new_text=$3
output_file="modified.txt"

sed "s/$old_text/$new_text/g" "$input_file" > "$output_file"
echo "Replacements completed.check the output files: $output_file"
```

Execute the script:



```
sreenath@LAPTOP-LH8NE438:~/file1/myscripts$ ./replace.sh original.txt old_te
xt new_text
```

After executions the replacement of the text:



```
sreenath@LAPTOP-LH8NE438:~/file1/myscripts$ cat modified.txt
This is some sample text file that contains new_text.
This is some sample text file that contains new_text format.
Here's another line with new_text apprearing twice.
new_text can be at the begining of the line as well.
there can be multiple line without new_text too.
this is the last line and this also has the new_text
```

To run this script, save it to a file (for example, script.sh), give it execute permissions using chmod +x script.sh, and then run it with three arguments: the input file, the old text to replace, and the new text. For example:

./script.sh input.txt old_text new_text

This will create a new file named output_input.txt with all occurrences of “old_text” replaced with “new_text”. Replace input.txt, old_text, and new_text with your actual file name and texts.