

Assignment Shell-Scripting

Que 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".

Ans:

```
file="myfile.txt"
```

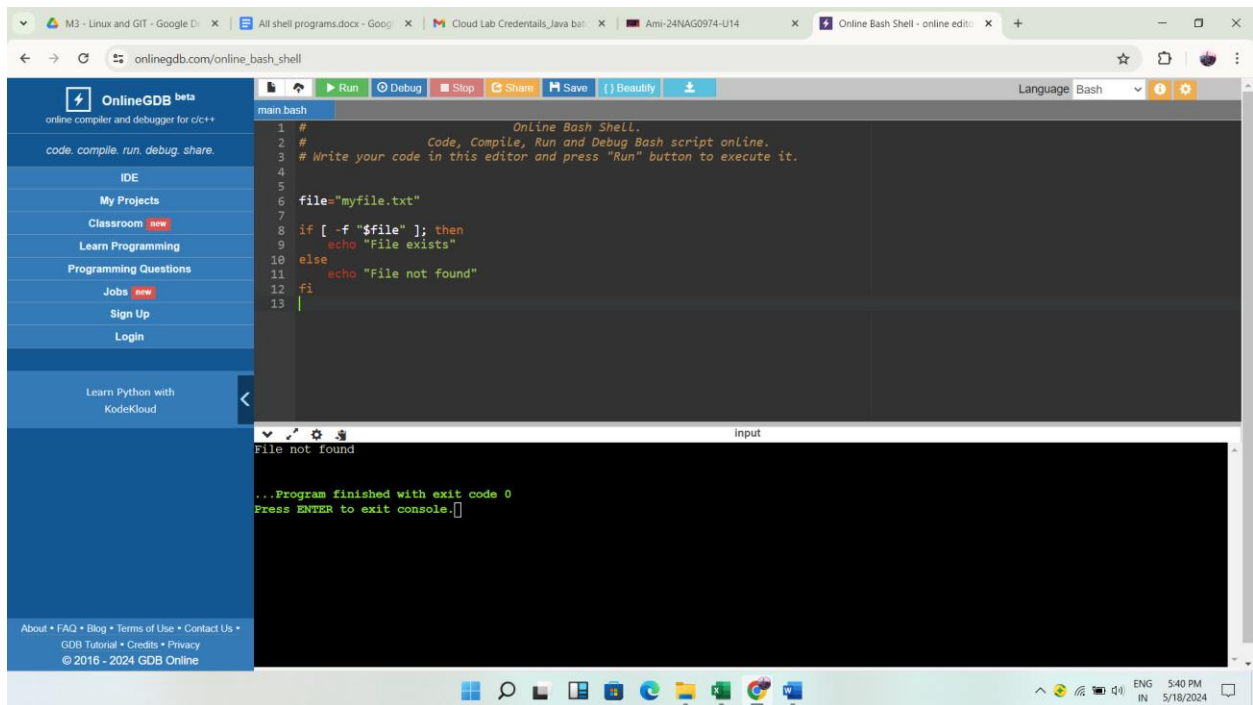
```
if [ -f "$file" ]; then
```

```
    echo "File exists"
```

```
else
```

```
    echo "File not found"
```

```
fi
```



The screenshot shows the OnlineGDB web interface. The code editor contains the following Bash script:

```
1 # Online Bash Shell.
2 # Code, Compile, Run and Debug Bash script online.
3 # Write your code in this editor and press "Run" button to execute it.
4
5
6 file="myfile.txt"
7
8 if [ -f "$file" ]; then
9     echo "File exists"
10
11 else
12     echo "File not found"
13 fi
```

The output console shows the result of the script execution:

```
File not found
...Program finished with exit code 0
Press ENTER to exit console.
```

Que 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.

Ans.

```
while true; do
```

```
    read -p "Enter a number (enter 0 to stop): " number
```

```
    if [ "$number" -eq 0 ]; then
```

```
        echo "Exiting..."
```

```
        break
```

```
    fi
```

```
    if [ $((number % 2)) -eq 0 ]; then
```

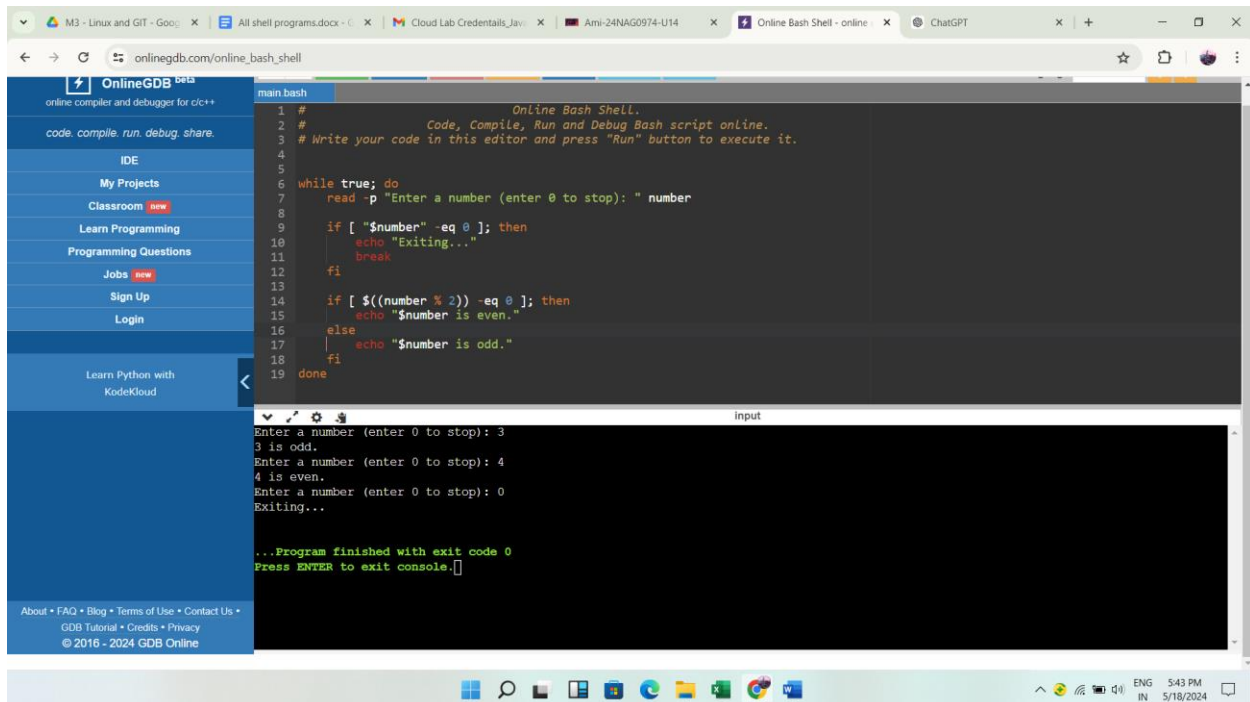
```
        echo "$number is even."
```

```
    else
```

```
        echo "$number is odd."
```

```
    fi
```

```
done
```



The screenshot shows the OnlineGDB web interface. The left sidebar contains navigation links: OnlineGDB (online compiler and debugger for c/c++), code, compile, run, debug, share, IDE, My Projects, Classroom, Learn Programming, Programming Questions, Jobs, Sign Up, Login, and Learn Python with KodeKloud. The main editor area displays a Bash script with line numbers 1 through 19. The script implements a loop that prompts the user for a number and checks if it is even or odd. The output console at the bottom shows the script's execution: it prompts for a number, receives 3 (odd), 4 (even), and 0 (exit), and finally prints "Exiting...". The status bar at the bottom indicates the program finished with exit code 0.

```
1 # Online Bash Shell.
2 # Code, Compile, Run and Debug Bash script online.
3 # Write your code in this editor and press "Run" button to execute it.
4
5
6 while true; do
7     read -p "Enter a number (enter 0 to stop): " number
8
9     if [ "$number" -eq 0 ]; then
10         echo "Exiting..."
11         break
12     fi
13
14     if [ $((number % 2)) -eq 0 ]; then
15         echo "$number is even."
16     else
17         echo "$number is odd."
18     fi
19 done
```

input

```
Enter a number (enter 0 to stop): 3
3 is odd.
Enter a number (enter 0 to stop): 4
4 is even.
Enter a number (enter 0 to stop): 0
Exiting...
...Program finished with exit code 0
Press ENTER to exit console.
```

Que 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.

Ans:

```
count_lines() {  
    wc -l "$1"  
}
```

```
filenames=("file1.txt" "file2.csv" "data.py")
```

```
for filename in "${filenames[@]}; do
```

```
    if [[ -f "$filename" ]]; then
```

```
        echo "Number of lines in $filename:"
```

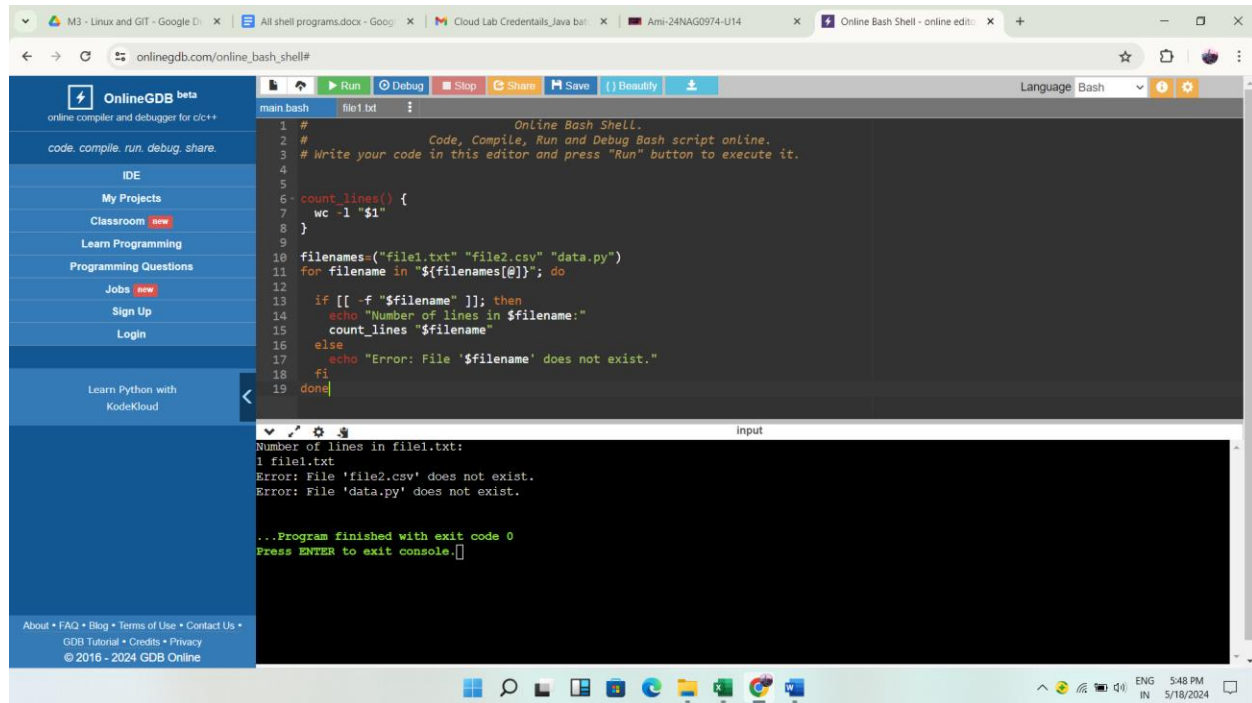
```
        count_lines "$filename"
```

```
    else
```

```
        echo "Error: File '$filename' does not exist."
```

```
    fi
```

```
done
```



Que 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").

Ans.

```
mkdir TestDir
```

```
for i in {1..10}
```

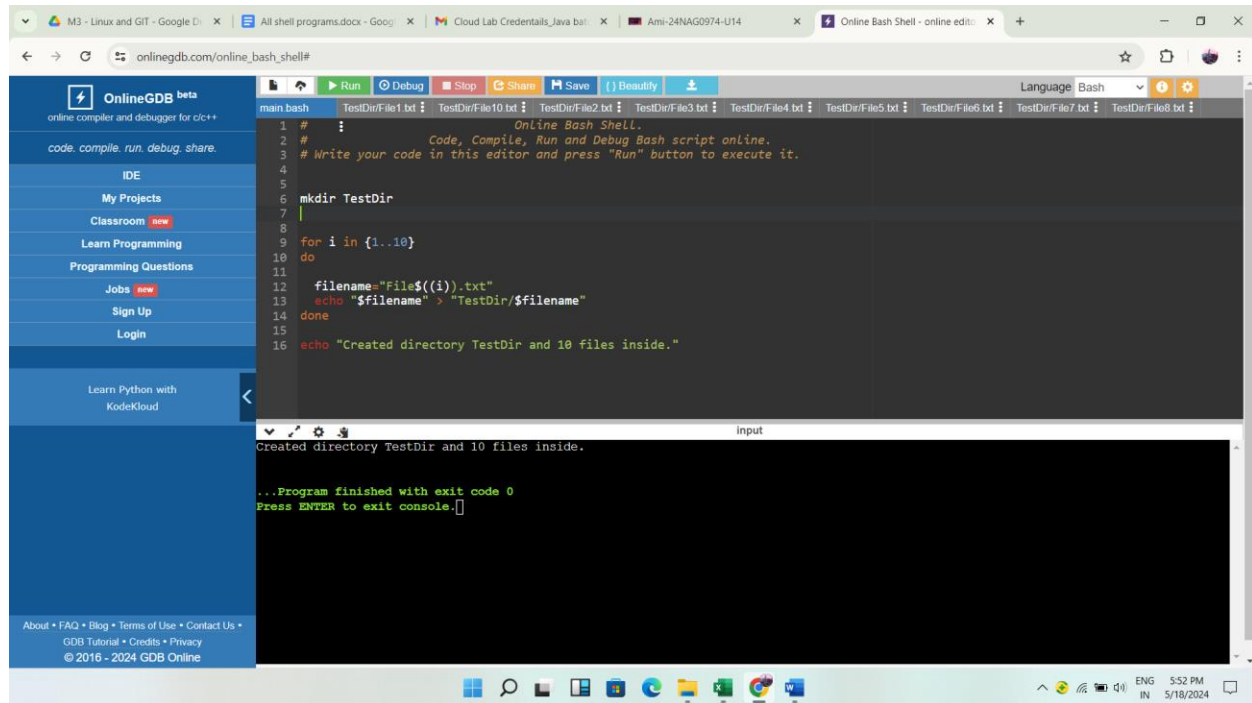
```
do
```

```
    filename="File${i}.txt"
```

```
    echo "$filename" > "TestDir/$filename"
```

```
done
```

```
echo "Created directory TestDir and 10 files inside."
```



Que 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.

Ans.

```
create_directory() {
```

```
    local directory="$1"
```

```
    local debug_mode="$2"
```

```
    if [[ ! -d "$directory" ]]; then
```

```
        mkdir -p "$directory"
```

```
        if [[ $? -eq 0 && $debug_mode -eq true ]]; then
```

```
            echo "Directory created: $directory"
```

```
        fi
```

```
    else
```

```
        if [[ $debug_mode -eq true ]]; then
```

```
            echo "Directory already exists: $directory"
```

```

fi

fi

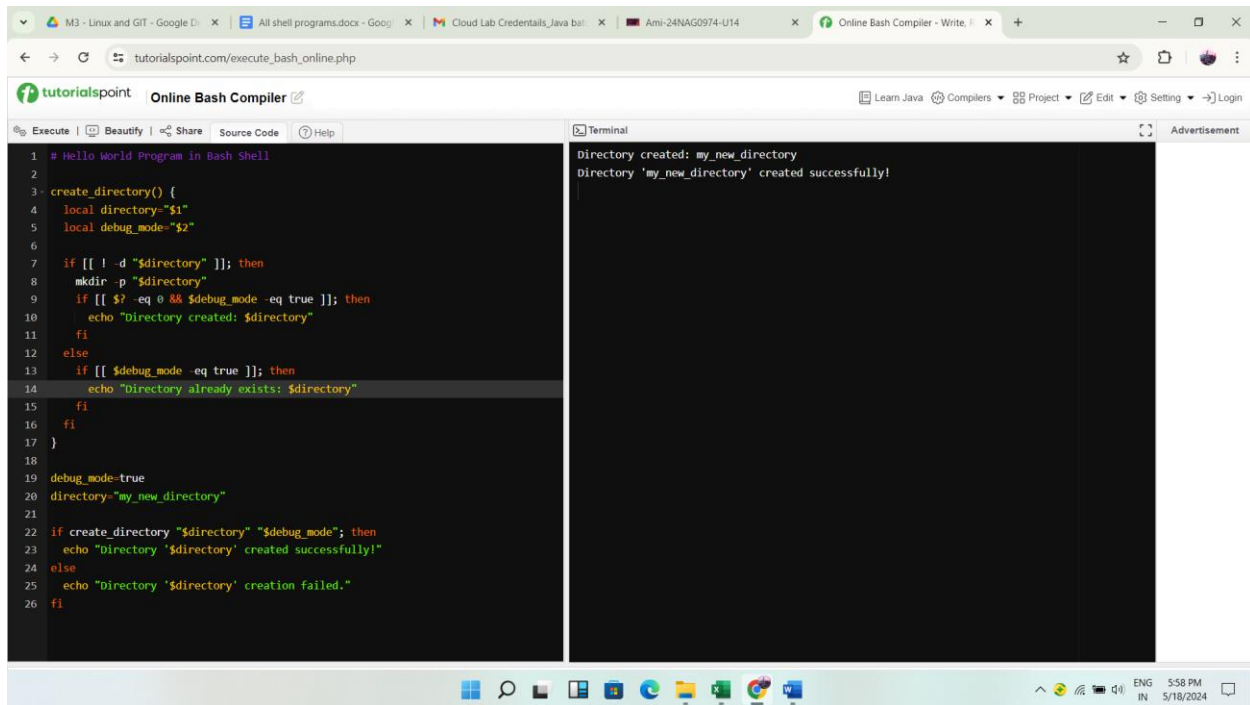
}

debug_mode=true

directory="my_new_directory"

if create_directory "$directory" "$debug_mode"; then
    echo "Directory '$directory' created successfully!"
else
    echo "Directory '$directory' creation failed."
fi

```



The screenshot shows the Online Bash Compiler interface. The left pane displays a Bash script with the following content:

```

1 # Hello World Program in Bash Shell
2
3 create_directory() {
4     local directory="$1"
5     local debug_mode="$2"
6
7     if [[ ! -d "$directory" ]]; then
8         mkdir -p "$directory"
9         if [[ $? -eq 0 && $debug_mode -eq true ]]; then
10             echo "Directory created: $directory"
11         fi
12     else
13         if [[ $debug_mode -eq true ]]; then
14             echo "Directory already exists: $directory"
15         fi
16     fi
17 }
18
19 debug_mode=true
20 directory="my_new_directory"
21
22 if create_directory "$directory" "$debug_mode"; then
23     echo "Directory '$directory' created successfully!"
24 else
25     echo "Directory '$directory' creation failed."
26 fi

```

The right pane shows the terminal output of the script execution:

```

Directory created: my_new_directory
Directory 'my_new_directory' created successfully!

```

Que. 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.

```
error_lines=$(grep "ERROR" "$logfile")
```

```
# Check if any errors were found
```

```
if [[ -z "$error_lines" ]]; then
```

```
    echo "No errors found in the log file."
```

```
    exit 0
```

```
fi
```

```
# Use awk to extract date, time, and error message
```

```
echo "Extracted Errors:"
```

```
echo "-----"
```

```
awk '
```

```
{
```

```
    # Split the line by spaces (assuming space separated format)
```

```
    split($0, fields, " ");
```

```
    # Extract date (assuming first 3 fields)
```

```
    date = fields[1]" "fields[2]" "fields[3];
```

```
    # Remove leading/trailing spaces from error message (using sed)
```

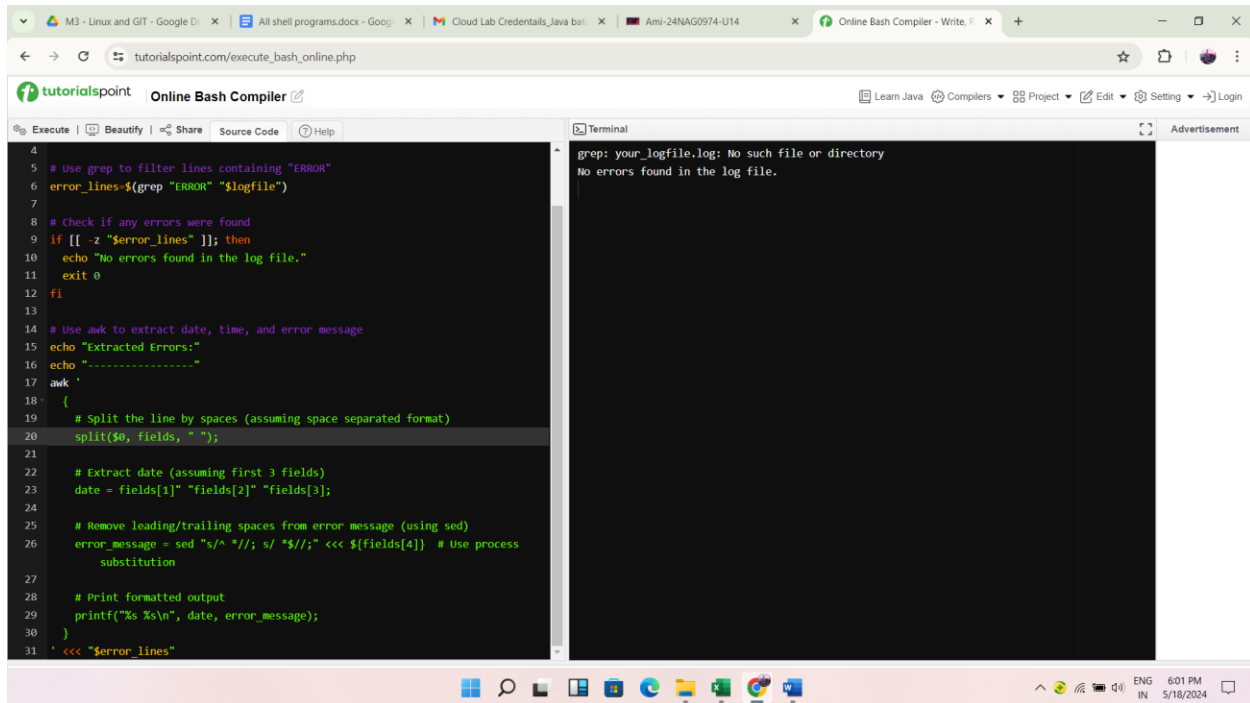
```
    error_message = sed "s/^ *//; s/ *$//;" <<< ${fields[4]} # Use process substitution
```

```
    # Print formatted output
```

```
    printf("%s %s\n", date, error_message);
```

```
}
```

```
' <<< "$error_lines"
```



The screenshot shows the Online Bash Compiler interface. The left pane contains a Bash script with the following content:

```
4
5 # Use grep to filter lines containing "ERROR"
6 error_lines=$(grep "ERROR" "$logfile")
7
8 # Check if any errors were found
9 if [[ -z "$error_lines" ]]; then
10     echo "No errors found in the log file."
11     exit 0
12 fi
13
14 # Use awk to extract date, time, and error message
15 echo "Extracted Errors:"
16 echo "-----"
17 awk '
18 {
19     # Split the line by spaces (assuming space separated format)
20     split($0, fields, " ");
21
22     # Extract date (assuming first 3 fields)
23     date = fields[1] " " fields[2] " " fields[3];
24
25     # Remove leading/trailing spaces from error message (using sed)
26     error_message = sed "s/^ *//; s/ *$//;" <<< ${fields[4]} # Use process
27     substitution
28
29     # Print formatted output
30     printf("%s %s\n", date, error_message);
31 }' <<< "$error_lines"
```

The right pane shows the terminal output:

```
grep: your_logfile.log: No such file or directory
No errors found in the log file.
```

Que 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.

Ans.

```
if [ $# -ne 2 ]; then
```

```
    echo "Usage: $0 <input_file> <new_text>"
```

```
    exit 1
```

```
fi
```

```
input_file="$1"
```

```
new_text="$2"
```

```
if [ ! -f "$input_file" ]; then
```

```
    echo "Error: File '$input_file' does not exist."
```

```
    exit 1
```

```
fi
```



```
output_file="${input_file%.txt}.replaced.txt"
```

```
sed -i "s/old_text/$new_text/g" "$input_file"
```

```
if [ $? -eq 0 ]; then
```

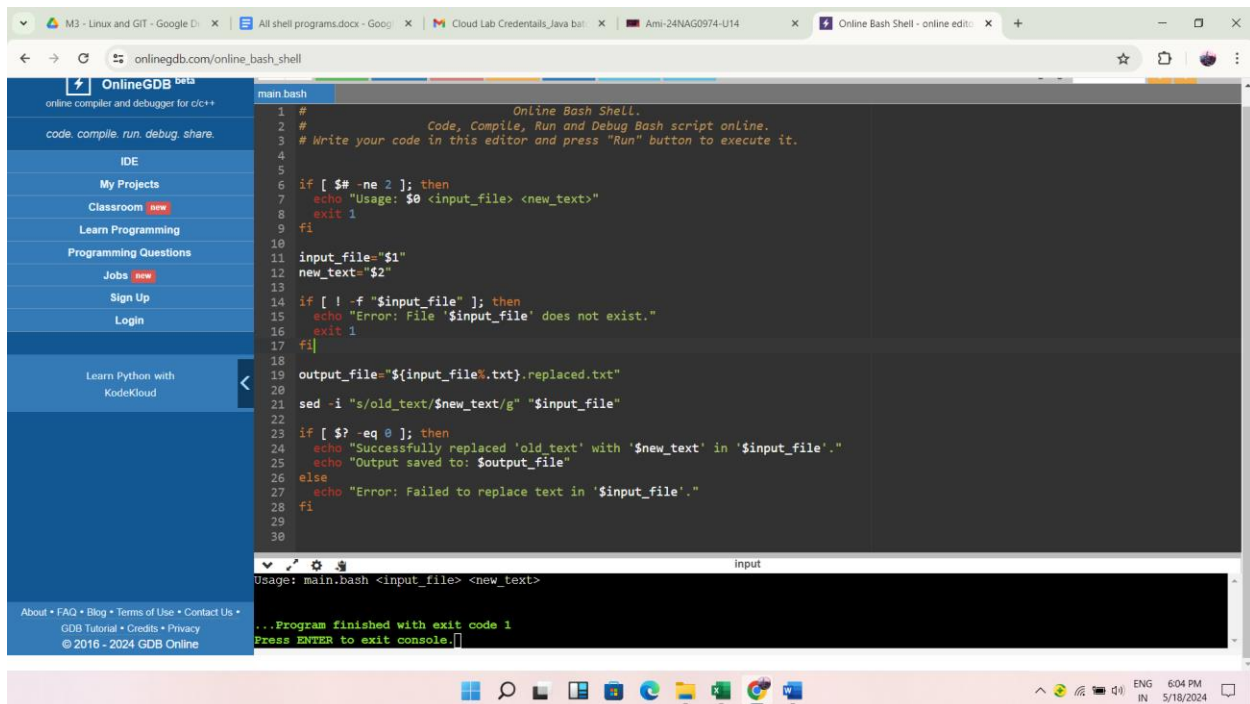
```
    echo "Successfully replaced 'old_text' with '$new_text' in '$input_file'."
```

```
    echo "Output saved to: $output_file"
```

```
else
```

```
    echo "Error: Failed to replace text in '$input_file'."
```

```
fi
```



The screenshot shows the OnlineGDB web interface. The browser tabs include 'M3 - Linux and GIT - Google D...', 'All shell programs.docx - Google...', 'Cloud Lab Credentials_Java bat...', 'Ami-24NAG0974-U14', and 'Online Bash Shell - online edit...'. The address bar shows 'onlinegdb.com/online_bash_shell'. The left sidebar contains navigation links: 'OnlineGDB beta', 'online compiler and debugger for c/c++', 'code, compile, run, debug, share.', 'IDE', 'My Projects', 'Classroom new', 'Learn Programming', 'Programming Questions', 'Jobs new', 'Sign Up', 'Login', 'Learn Python with KodeKloud', and footer links 'About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy © 2016 - 2024 GDB Online'. The main editor area shows a Bash script with line numbers 1 to 30. The script implements a text replacement function using 'sed'. The terminal output at the bottom shows the usage message: 'Usage: main.bash <input_file> <new_text>' and a message indicating the program finished with exit code 1.

```
1 # Online Bash Shell.
2 # Code, Compile, Run and Debug Bash script online.
3 # Write your code in this editor and press "Run" button to execute it.
4
5
6 if [ $# -ne 2 ]; then
7     echo "Usage: $0 <input_file> <new_text>"
8     exit 1
9 fi
10
11 input_file="$1"
12 new_text="$2"
13
14 if [ ! -f "$input_file" ]; then
15     echo "Error: File '$input_file' does not exist."
16     exit 1
17 fi
18
19 output_file="${input_file%.txt}.replaced.txt"
20
21 sed -i "s/old_text/$new_text/g" "$input_file"
22
23 if [ $? -eq 0 ]; then
24     echo "Successfully replaced 'old_text' with '$new_text' in '$input_file'."
25     echo "Output saved to: $output_file"
26 else
27     echo "Error: Failed to replace text in '$input_file'."
28 fi
29
30
```

Usage: main.bash <input_file> <new_text>

...Program finished with exit code 1
Press ENTER to exit console.[]