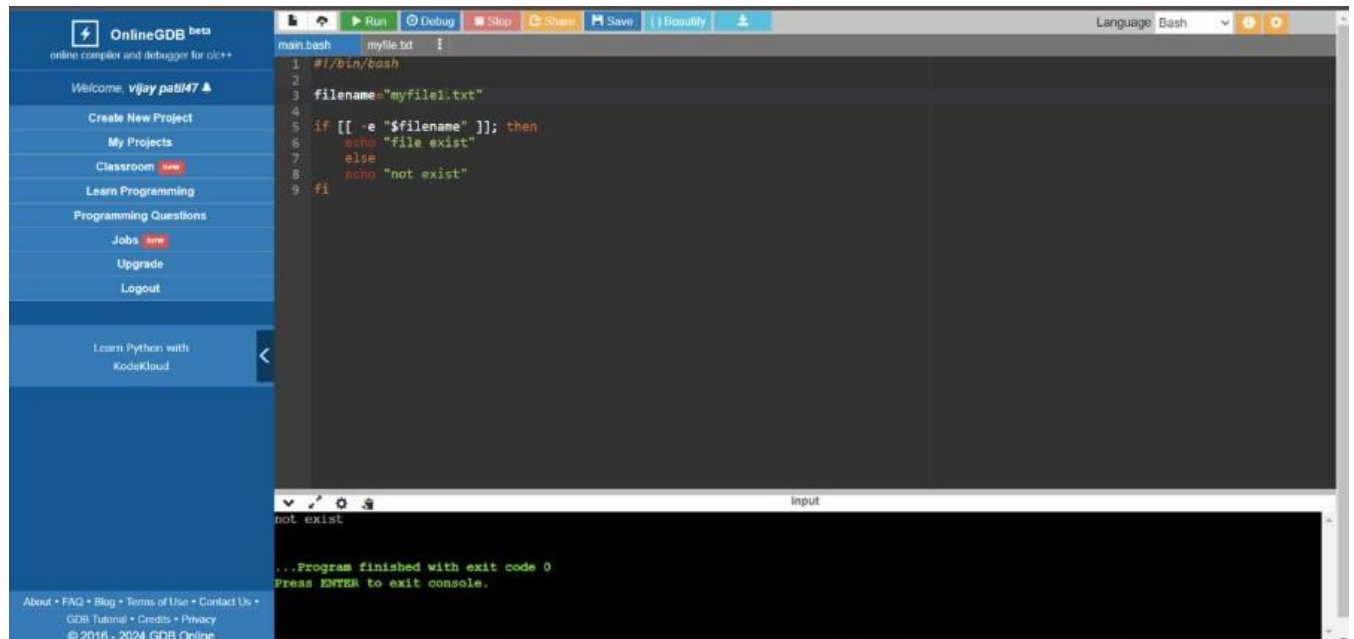


Assignment 1

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

Input and Output Snapshot:-



The screenshot shows the OnlineGDB interface. On the left is a sidebar with navigation links. The main editor displays a Bash script in a file named 'myfile.txt'. The script checks for the existence of 'myfile1.txt'. The output console at the bottom shows the result of running the script.

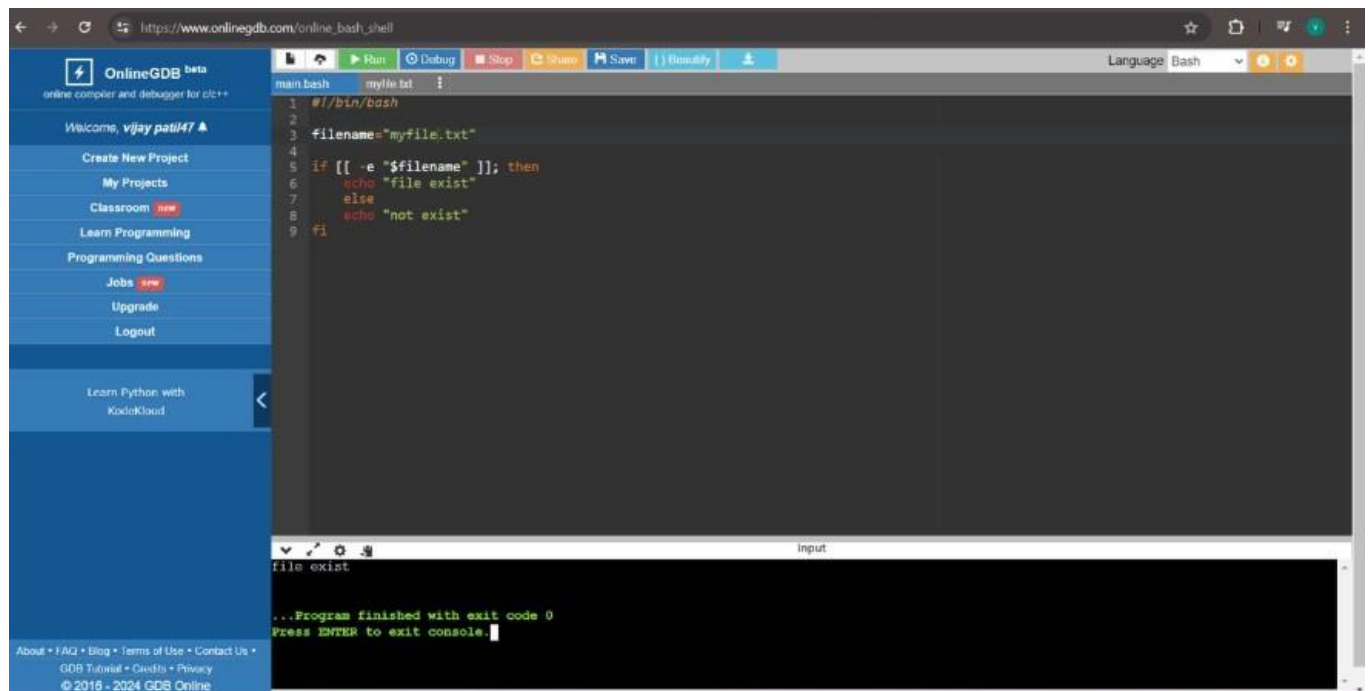
```
1 #!/bin/bash
2
3 filename="myfile1.txt"
4
5 if [[ -e "$filename" ]]; then
6     echo "file exist"
7 else
8     echo "not exist"
9 fi
```

not exist

...Program finished with exit code 0
Press ENTER to exit console.

OUTPUT:

File not exist



The screenshot shows the OnlineGDB web interface. On the left is a sidebar with navigation links like 'Create New Project', 'My Projects', 'Classroom', 'Learn Programming', 'Programming Questions', 'Jobs', 'Upgrade', 'Logout', and 'Learn Python with KodeKloud'. The main area displays a Bash script in a dark-themed editor. The script sets a variable 'filename' to 'myfile.txt' and uses an 'if' statement to check if the file exists. If it does, it prints 'file exist'. The output window at the bottom shows the result of the script execution: 'file exist' and a confirmation message that the program finished with exit code 0.

```
1 #!/bin/bash
2
3 filename="myfile.txt"
4
5 if [[ -e "$filename" ]]; then
6     echo "file exist"
7 else
8     echo "not exist"
9 fi
```

file exist

...Program finished with exit code 0
Press ENTER to exit console.

OUTPUT:

File exist

2] Write a script that reads number 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 -p "Enter a number (enter 0 to quit): " num
```

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

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

```
break
```

```
fi
```

```
if (( num % 2 == 0 )); then
```

```
echo "$num is even."

else echo "$num is odd."

fi

done
```

OUTPUT:



```
input
Enter a number (enter 0 to stop): 10
10 is even.
Enter a number (enter 0 to stop): 9
9 is odd.
Enter a number (enter 0 to stop): 13
13 is odd.
Enter a number (enter 0 to stop): 23
23 is odd.
Enter a number (enter 0 to stop): 0
Exiting program...

...Program finished with exit code 0
Press ENTER to exit console.
```

3] Create a function that takes a filename as an argument and print the number of lines in the files. Call this function from your script with different filenames.

Nano count_line.sh

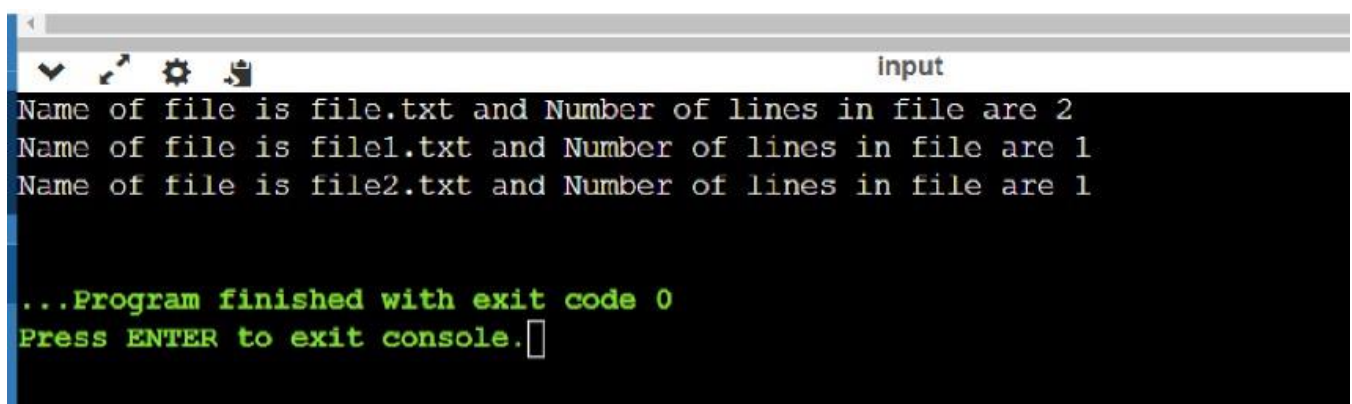
This command is use for the creating script file.

Script:

```
#!/bin/bash
```

```
count_lines() {  
    local filename=$1  
    if [-f "$filename"]; then  
        local line_count= $(wc -l <"$filename")  
        echo "Name of file is "$filename and Number of lines in file are $lines"  
    else  
        echo "Error: The file '$filename' does not exist."  
    fi  
}  
  
count_lines "file1.txt"  
count_lines "file2.txt"  
count_lines "file3.txt"
```

OUTPUT:



The screenshot shows a terminal window titled "input". The output of the script is displayed in a monospaced font. The first three lines show the script being run on "file.txt", "file1.txt", and "file2.txt", each reporting the number of lines. The final two lines show the program finishing with exit code 0 and a prompt to press ENTER to exit the console.

```
input  
Name of file is file.txt and Number of lines in file are 2  
Name of file is file1.txt and Number of lines in file are 1  
Name of file is file2.txt and Number of lines in file are 1  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

4] Write a script that creates a directory named TestDir and inside it, create ten files named File1.txt, File2.txt,.... File 10.txt. Each file should contain its filename as its content(e.g., File1.txt contains "File1.txt").

Script:

```
For i in (1...10); do
```

```
    F= "File$i.txt"
```

```
    Echo $f> "$f"
```

```
done
```

```
echo "file created."
```



```
dir_name= "TestDir"
```

```
mkdir -p "$dir_name"
```

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

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

```
    filepath= "$dir_name/$filename"
```

```
echo "$filename" > "$filepath"
```

```
done
```

```
echo "Directory '$dir_name' and files created successfully!"
```

5] Modify the script to handle errors, such as the directory already existing or lacking permission to create files.

```
#!/bin/bash
```

```
handle_error() {
```

```
    echo "Error: $1"
```

```
    exit 1
```

```
}
```

```
if [ -d "TestDir" ]; then
```

```
    handle_error "Directory 'TestDir' already exists."
```

```
fi
```

```
mkdir TestDir || handle_error "Failed to create directory 'TestDir'."
```

```
cd TestDir || handle_error "Failed to change into directory 'TestDir'."
```

```
for ((i=1; i<=10; i++)); do
```

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

```
    echo "$filename" > "$filename" || handle_error "Failed to create file  
'$filename'."
```

```
done
```

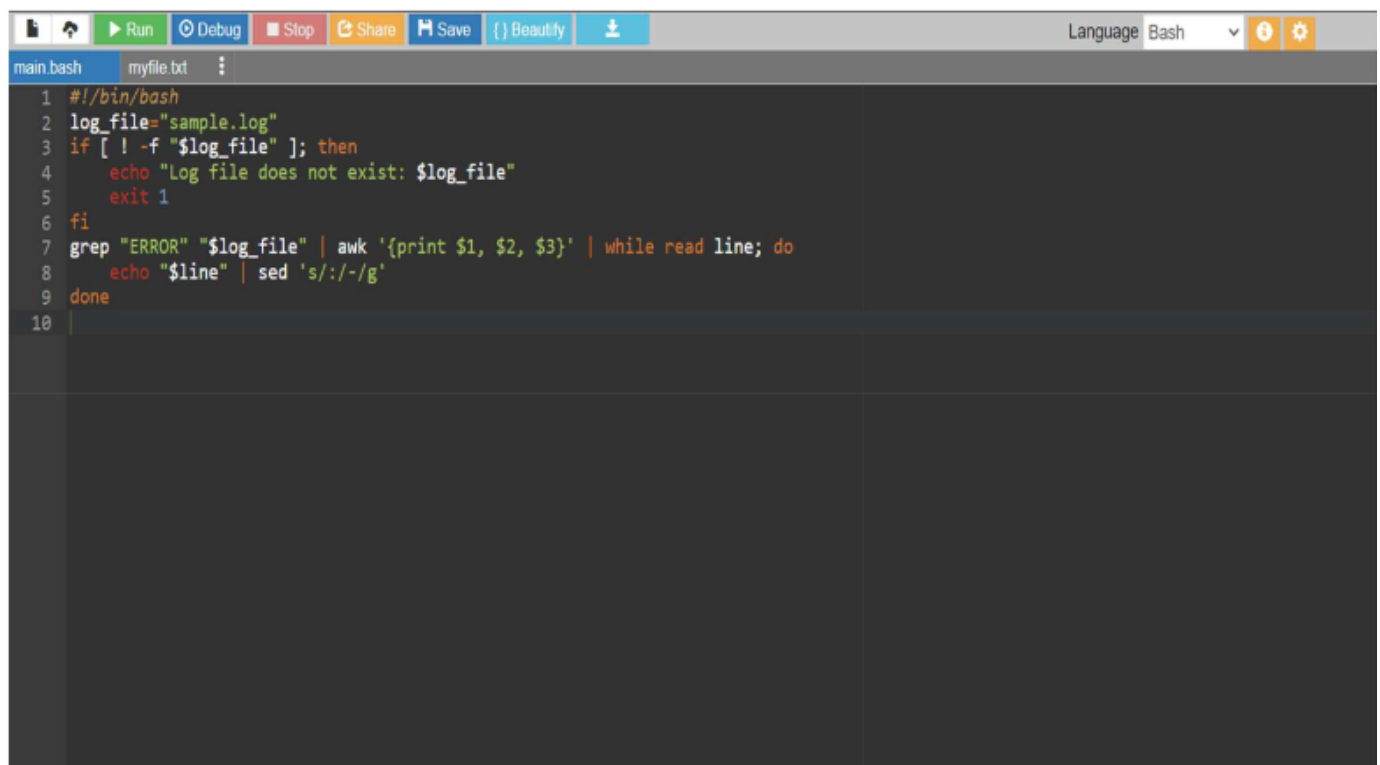
```
# Display message
```

```
echo "Files created successfully in TestDir."
```

directory and files created(or already exists).

...Program finished with exit code 0
Press ENTER to exit console.

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.



```
1 #!/bin/bash
2 log_file="sample.log"
3 if [ ! -f "$log_file" ]; then
4     echo "Log file does not exist: $log_file"
5     exit 1
6 fi
7 grep "ERROR" "$log_file" | awk '{print $1, $2, $3}' | while read line; do
8     echo "$line" | sed 's/:/-/g'
9 done
10
```

Script:

```
#!/bin/bash

log_file= "sample.log"

if[ ! -f "$log_file"]; then

    echo "Log file does not exist: $log_file"

    exit 1

fi

grep "ERROR" "$log_file" | awk '{print $1,$2,$3}' | while read line;

do

    echo "$line" | sed 's:/-/g'

done
```

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.

```
#!/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="$3"
output_file= "${input_file%.txt}_modified.txt"
if [!-f "$input_file"]; then
echo "error: file does not exist-$input_file"
exit 1
fi
sed "s/$old_text/$new_text/g" "$input_file"> "$output_file"
echo "Operation completed. Modified file is $output_file"
```

```
main.bash  myfile.txt  ⋮
1  #!/bin/bash
2
3  # Check for proper usage
4  if [ $# -ne 3 ]; then
5      echo "Usage: $0 input_file old_text new_text"
6      exit 1
7  fi
8
9  # Assign script arguments to variables
10 input_file="$1"
11 old_text="$2"
12 new_text="$3"
13 output_file="${input_file%.txt}_modified.txt" # Appends '_modified' to the original filename
14
15 # Check if the input file exists
16 if [ ! -f "$input_file" ]; then
17     echo "Error: File does not exist - $input_file"
18     exit 1
19 fi
20
21 # Use sed to replace all occurrences of old_text with new text
22 sed "s/$old_text/$new_text/g" "$input_file" > "$output_file"
23
24 echo "Operation completed. Modified file is $output_file"
25
```



28°C N