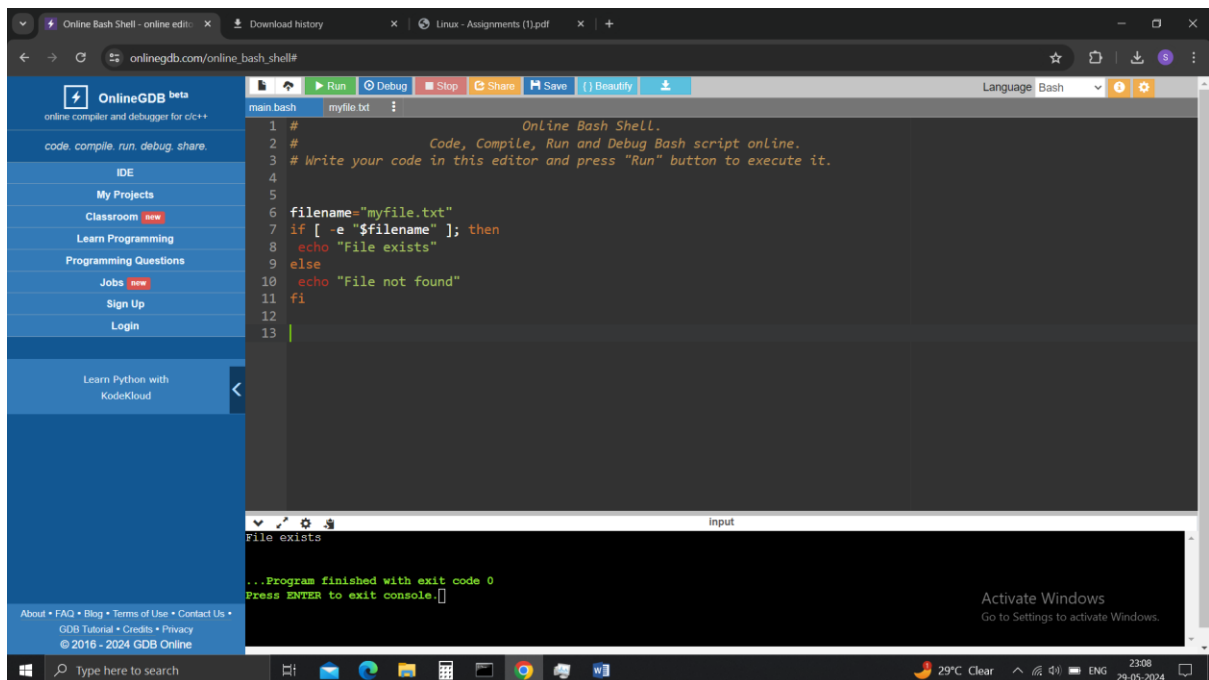


Task 1: Ensure the script checks if a specific file (e.g., myfile.txt) exists in the current directory. If it exists, print "File exists", otherwise print "File not found".



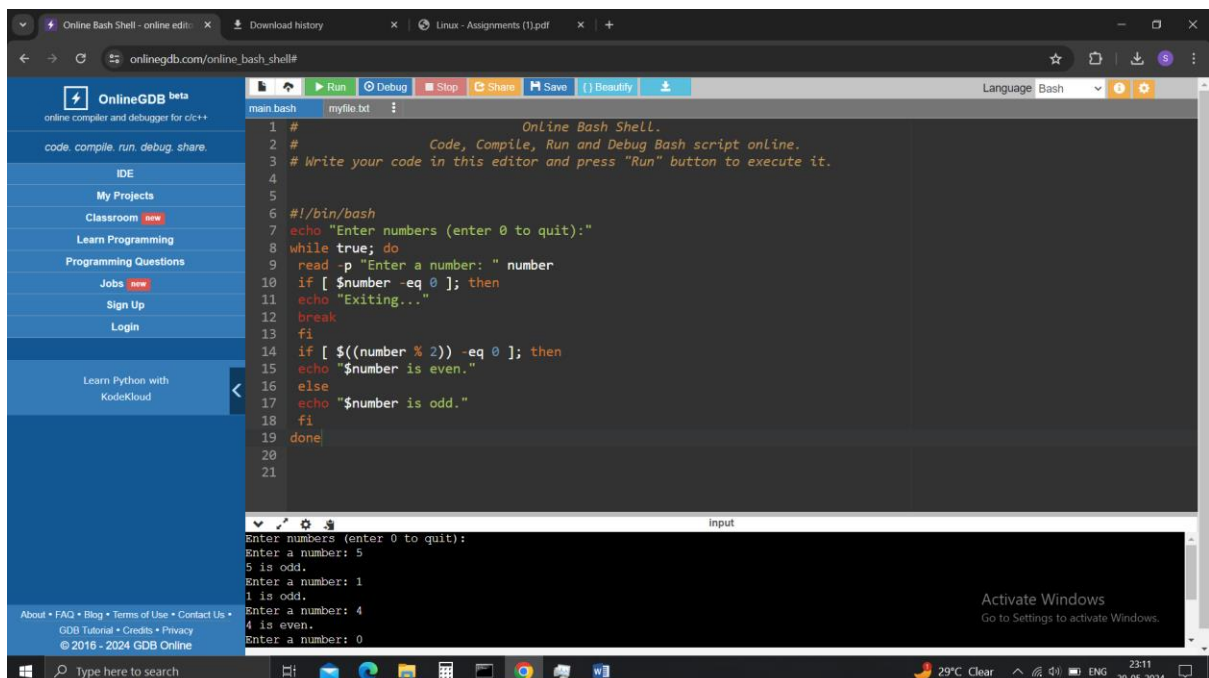
The screenshot shows the OnlineGDB interface with a Bash script in a file named 'myfile.txt'. The script checks if the file 'myfile.txt' exists in the current directory. If it exists, it prints 'File exists'; otherwise, it prints 'File not found'. The script has been executed, and the output 'File exists' is visible in the console. The console also shows the program finished with exit code 0 and a prompt to press ENTER to exit the console.

```
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 filename="myfile.txt"
7 if [ -e "$filename" ]; then
8     echo "File exists"
9 else
10    echo "File not found"
11 fi
12
13
```

File exists

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

Task 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.

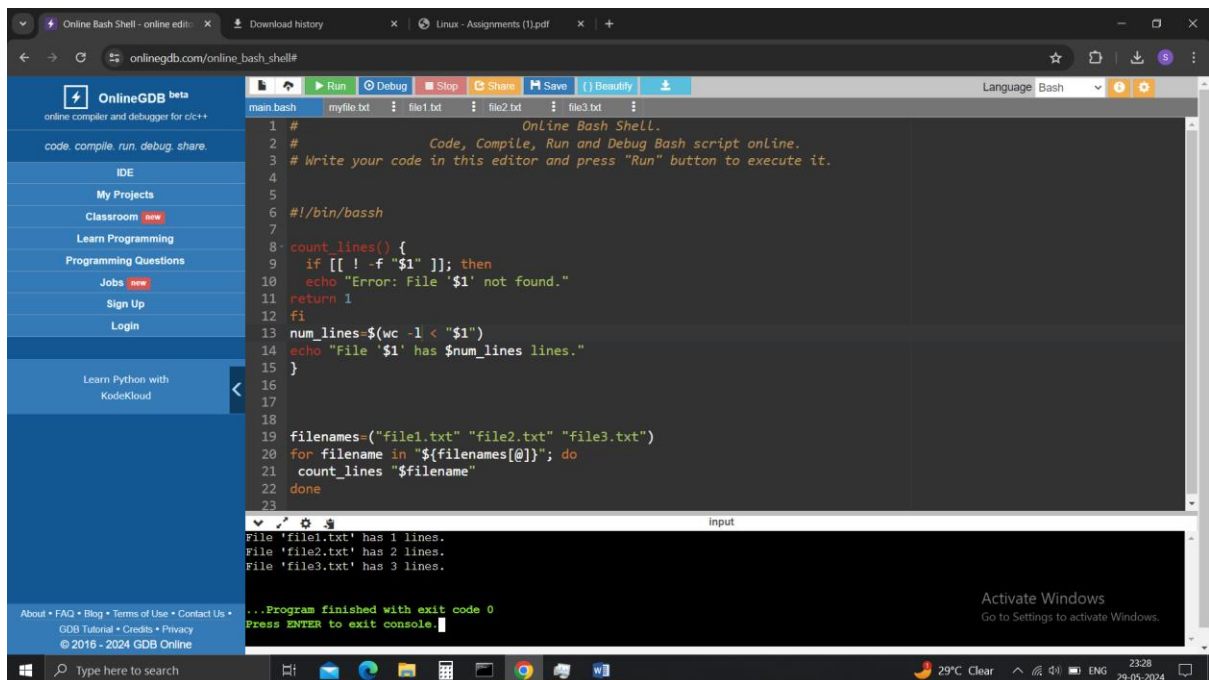


The screenshot shows the OnlineGDB interface with a Bash script in a file named 'myfile.txt'. The script reads numbers from the user until they enter '0'. For each number entered, it checks if it is odd or even and prints the result. The script has been executed, and the output shows the user entering numbers 5, 1, 4, and 0, with corresponding odd/even checks. The console also shows the program finished with exit code 0 and a prompt to press ENTER to exit the console.

```
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 #!/bin/bash
7 echo "Enter numbers (enter 0 to quit):"
8 while true; do
9     read -p "Enter a number: " number
10    if [ $number -eq 0 ]; then
11        echo "Exiting..."
12        break
13    fi
14    if [ $((number % 2)) -eq 0 ]; then
15        echo "$number is even."
16    else
17        echo "$number is odd."
18    fi
19 done
20
21
```

Enter numbers (enter 0 to quit):
Enter a number: 5
5 is odd.
Enter a number: 1
1 is odd.
Enter a number: 4
4 is even.
Enter a number: 0

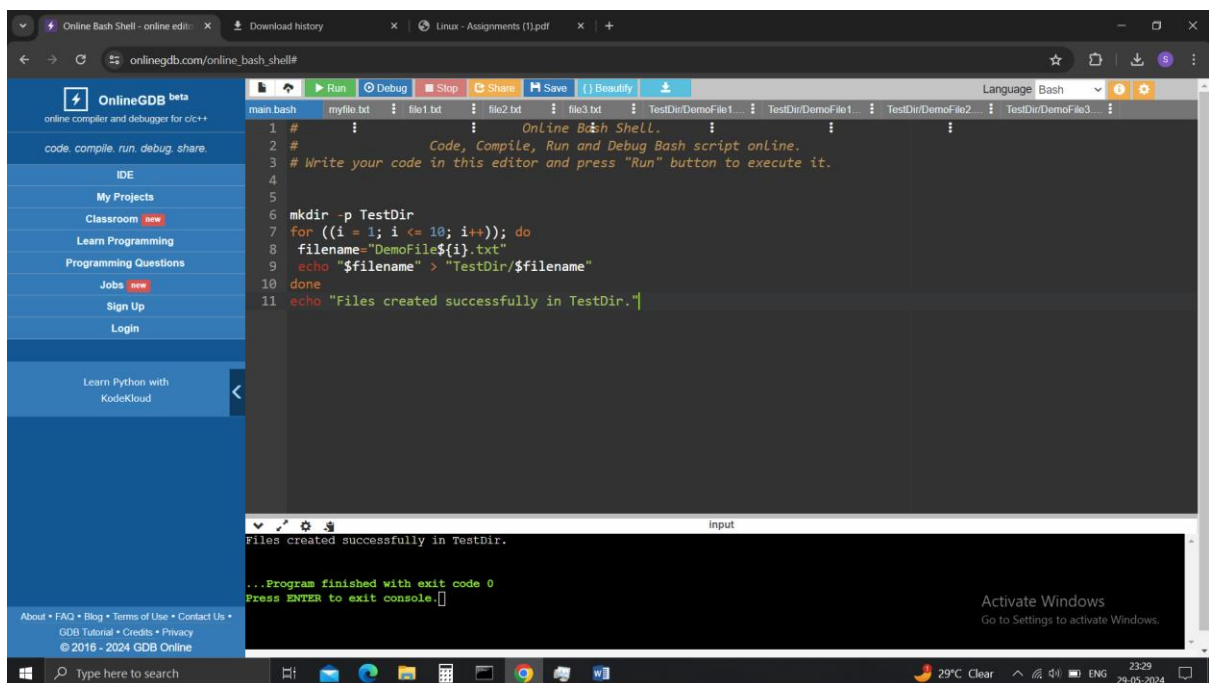
Task 3: Create a function that takes a filename as an argument and prints the number of lines in the file. Call this function from your script with different filenames.



The screenshot shows the OnlineGDB interface with a Bash script for Task 3. The script defines a function `count_lines()` that takes a filename as an argument and prints the number of lines in the file. It then calls this function for three files: `file1.txt`, `file2.txt`, and `file3.txt`. The output shows the number of lines for each file: 1, 2, and 3 respectively.

```
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 #!/bin/bash
6
7
8 count_lines() {
9     if [[ ! -f "$1" ]]; then
10         echo "Error: File '$1' not found."
11     fi
12     return 1
13     num_lines=$(wc -l < "$1")
14     echo "File '$1' has $num_lines lines."
15 }
16
17
18 filenames=("file1.txt" "file2.txt" "file3.txt")
19 for filename in "${filenames[@]}; do
20     count_lines "$filename"
21 done
22
23
24 File 'file1.txt' has 1 lines.
25 File 'file2.txt' has 2 lines.
26 File 'file3.txt' has 3 lines.
27
28 ...Program finished with exit code 0
29 Press ENTER to exit console.
```

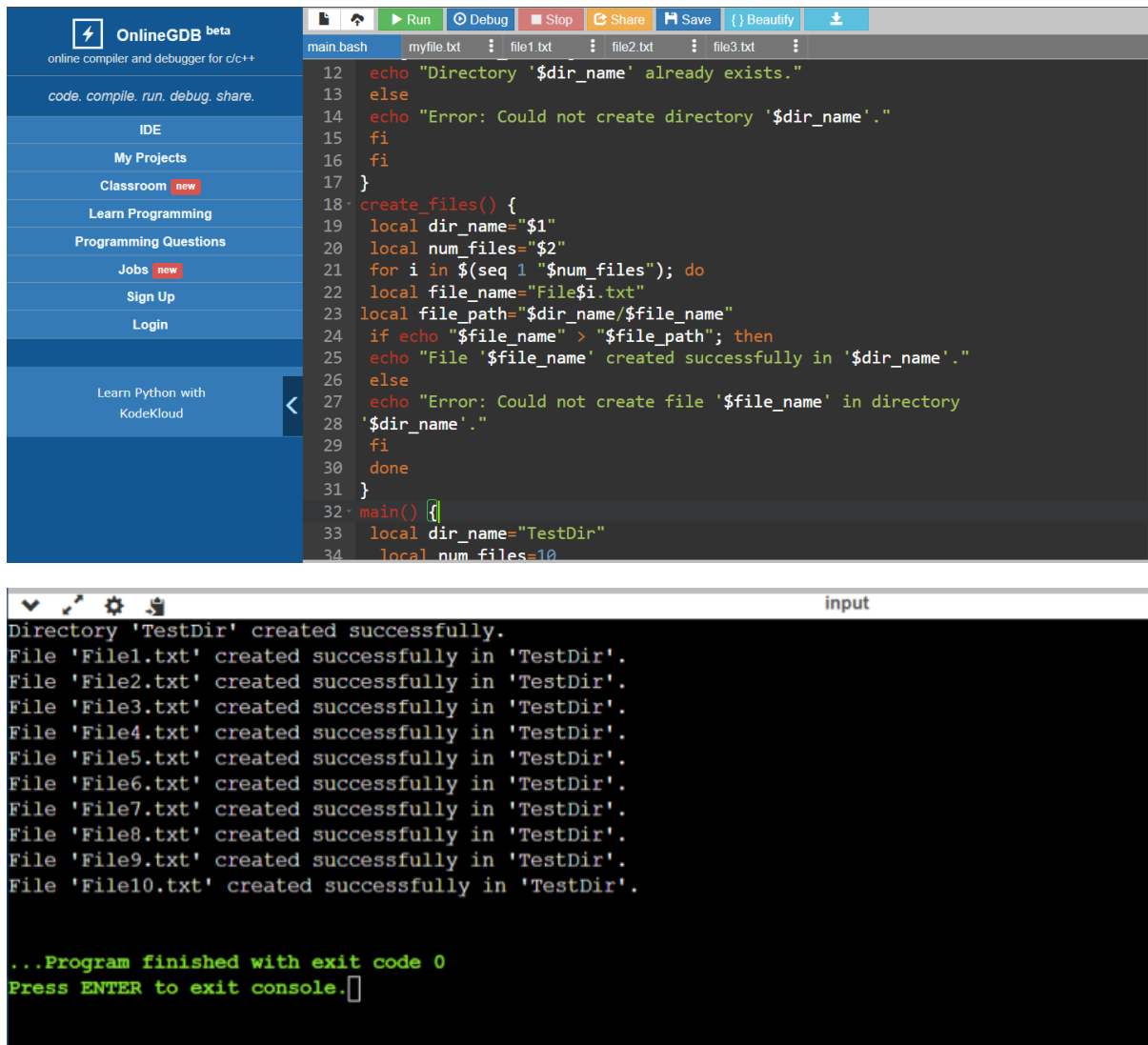
Task 4: Write a script that creates a directory named TestDir and inside it, creates ten files named File1.txt, File2.txt, ... File10.txt. Each file should contain its filename as its content (e.g., File1.txt contains "File1.txt").



The screenshot shows the OnlineGDB interface with a Bash script for Task 4. The script creates a directory named `TestDir` and then creates ten files named `File1.txt` through `File10.txt` inside it. Each file contains its filename as its content. The output shows the files created successfully in `TestDir`.

```
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 mkdir -p TestDir
7 for ((i = 1; i <= 10; i++)); do
8     filename="DemoFile${i}.txt"
9     echo "$filename" > "TestDir/$filename"
10 done
11 echo "Files created successfully in TestDir."
12
13
14 Files created successfully in TestDir.
15
16 ...Program finished with exit code 0
17 Press ENTER to exit console.
```

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



The screenshot shows the OnlineGDB IDE interface. The left sidebar contains navigation links: OnlineGDB beta, code. compile. run. debug. share., IDE, My Projects, Classroom new, Learn Programming, Programming Questions, Jobs new, Sign Up, Login, and Learn Python with KodeKloud. The main editor displays a Bash script with the following code:

```

12 echo "Directory '$dir_name' already exists."
13 else
14 echo "Error: Could not create directory '$dir_name'."
15 fi
16 fi
17 }
18 create_files() {
19 local dir_name="$1"
20 local num_files="$2"
21 for i in $(seq 1 "$num_files"); do
22 local file_name="File$i.txt"
23 local file_path="$dir_name/$file_name"
24 if echo "$file_name" > "$file_path"; then
25 echo "File '$file_name' created successfully in '$dir_name'."
26 else
27 echo "Error: Could not create file '$file_name' in directory
28 '$dir_name'."
29 fi
30 done
31 }
32 main() {
33 local dir_name="TestDir"
34 local num_files=10

```

The output window at the bottom shows the execution results:

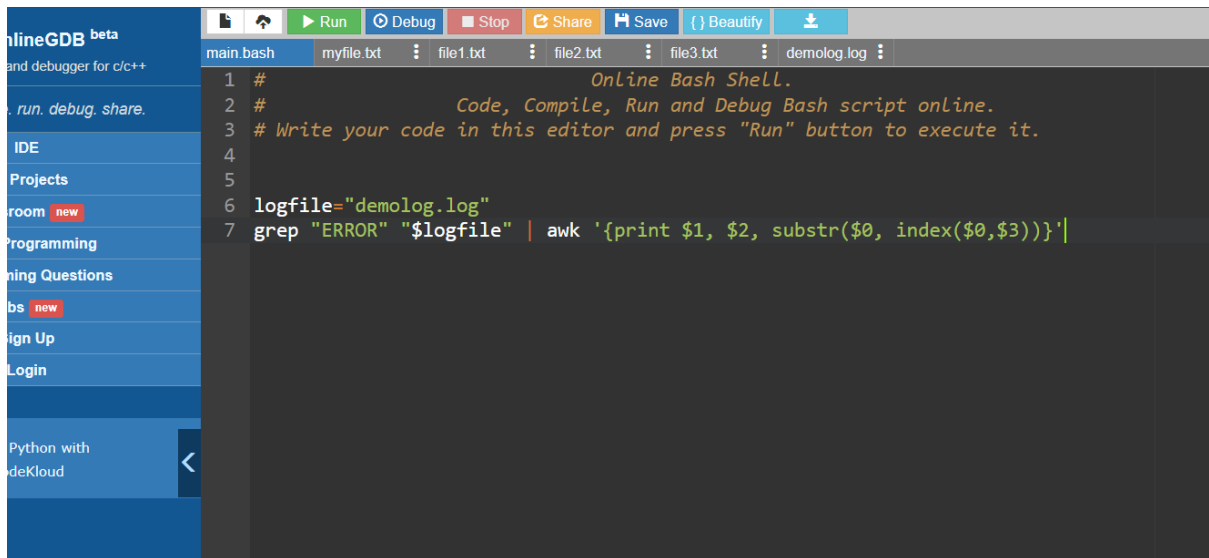
```

Directory 'TestDir' created successfully.
File 'File1.txt' created successfully in 'TestDir'.
File 'File2.txt' created successfully in 'TestDir'.
File 'File3.txt' created successfully in 'TestDir'.
File 'File4.txt' created successfully in 'TestDir'.
File 'File5.txt' created successfully in 'TestDir'.
File 'File6.txt' created successfully in 'TestDir'.
File 'File7.txt' created successfully in 'TestDir'.
File 'File8.txt' created successfully in 'TestDir'.
File 'File9.txt' created successfully in 'TestDir'.
File 'File10.txt' created successfully in 'TestDir'.

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

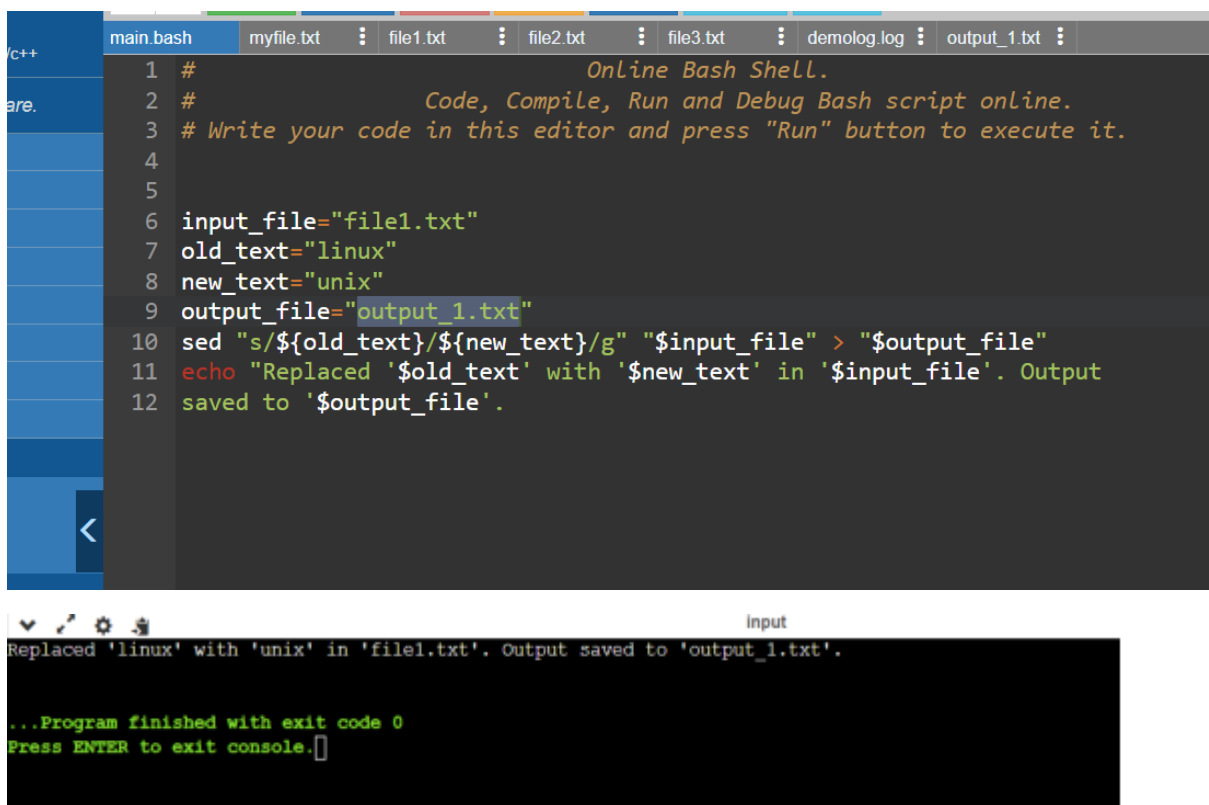
```

Task 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



```
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 logfile="demolog.log"
7 grep "ERROR" "$logfile" | awk '{print $1, $2, substr($0, index($0,$3))}'
```

Task 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.



```
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 input_file="file1.txt"
7 old_text="linux"
8 new_text="unix"
9 output_file="output_1.txt"
10 sed "s/${old_text}/${new_text}/g" "$input_file" > "$output_file"
11 echo "Replaced '$old_text' with '$new_text' in '$input_file'. Output
12 saved to '$output_file'."
```

Replaced 'linux' with 'unix' in 'file1.txt'. Output saved to 'output_1.txt'.

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