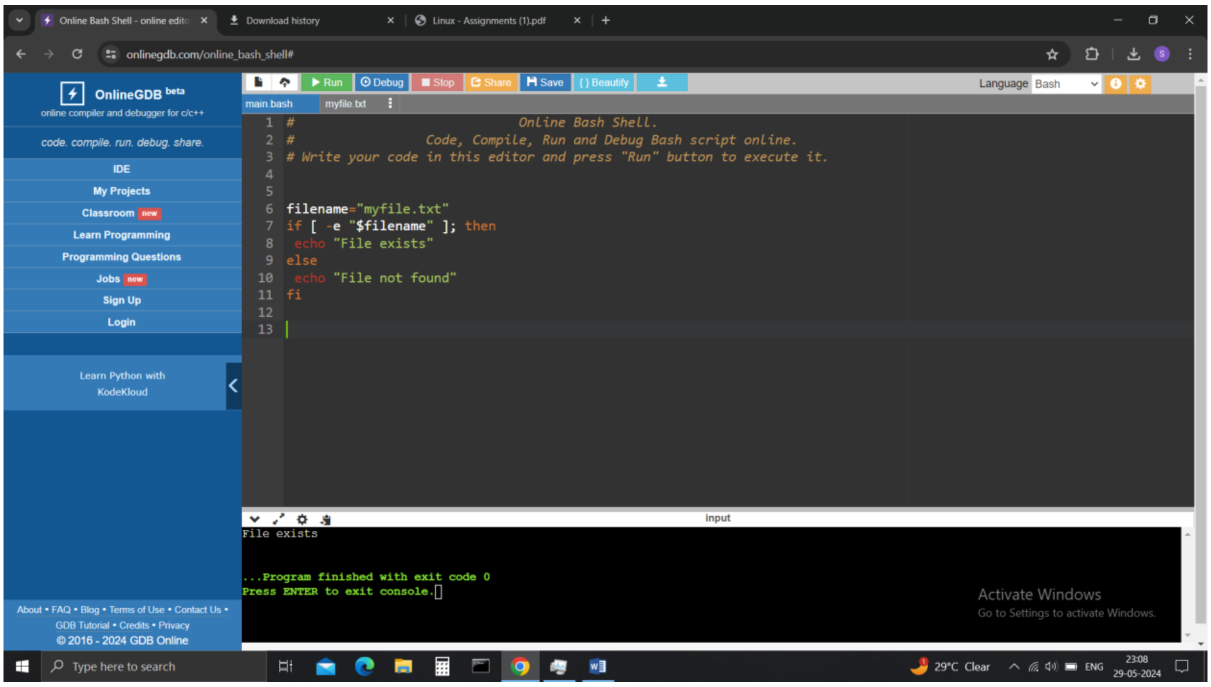
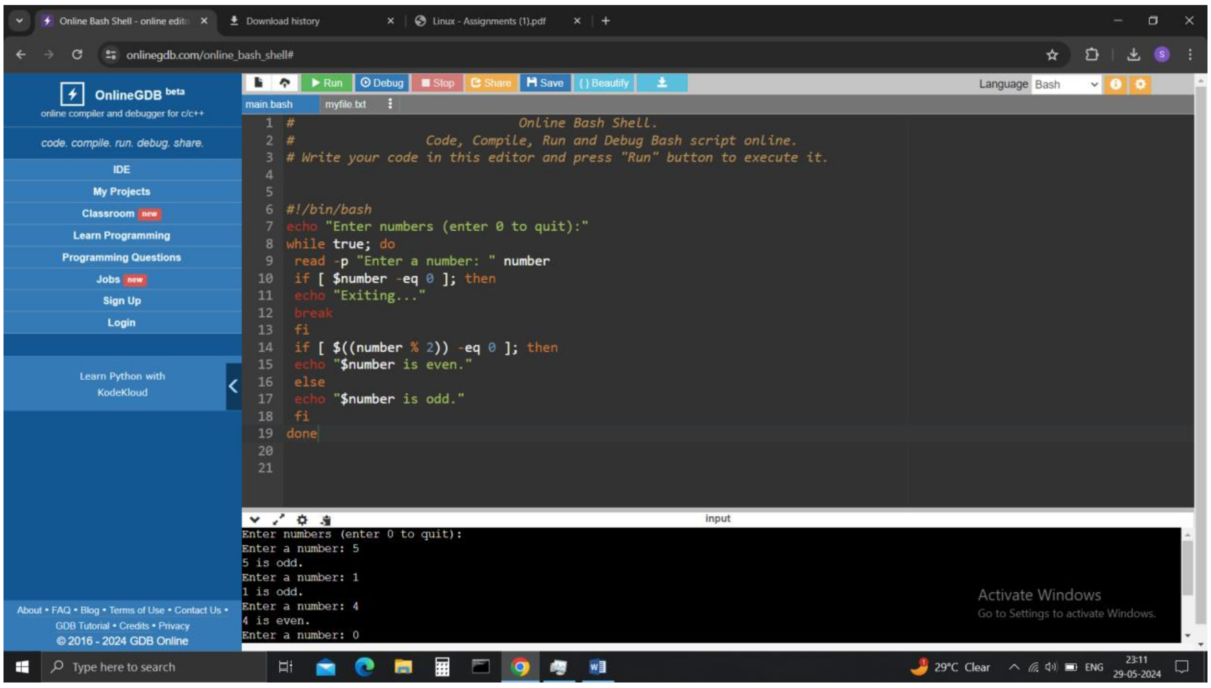
Task 1:Ensure the script checks if a specific file (e.g., myfile.txt) exists in the current directory. If it exists, print "File exists", otherwise print "File not found".



```
#                            Online Bash Shell.
#                Code, Compile, Run and Debug Bash script online.
# Write your code in this editor and press "Run" button to execute it.


filename="myfile.txt"
if [ -e "$filename" ]; then
 echo "File exists"
else
 echo "File not found"
fi

```

```
File exists


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

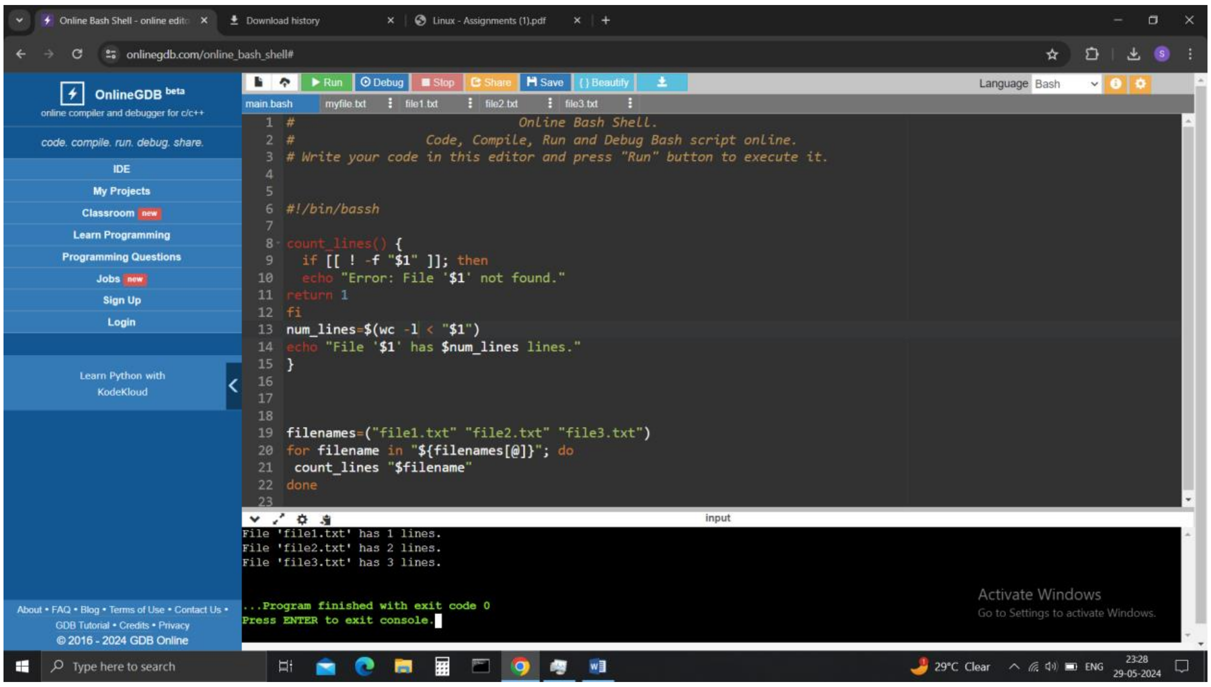
Task 2: Write a script that reads numbers from the user until they enter '0'. The script should also print whether each number is odd or even.



```
#                            Online Bash Shell.
#                Code, Compile, Run and Debug Bash script online.
# Write your code in this editor and press "Run" button to execute it.


#!/bin/bash
echo "Enter numbers (enter 0 to quit):"
while true; do
 read -p "Enter a number: " 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
```

```
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
```

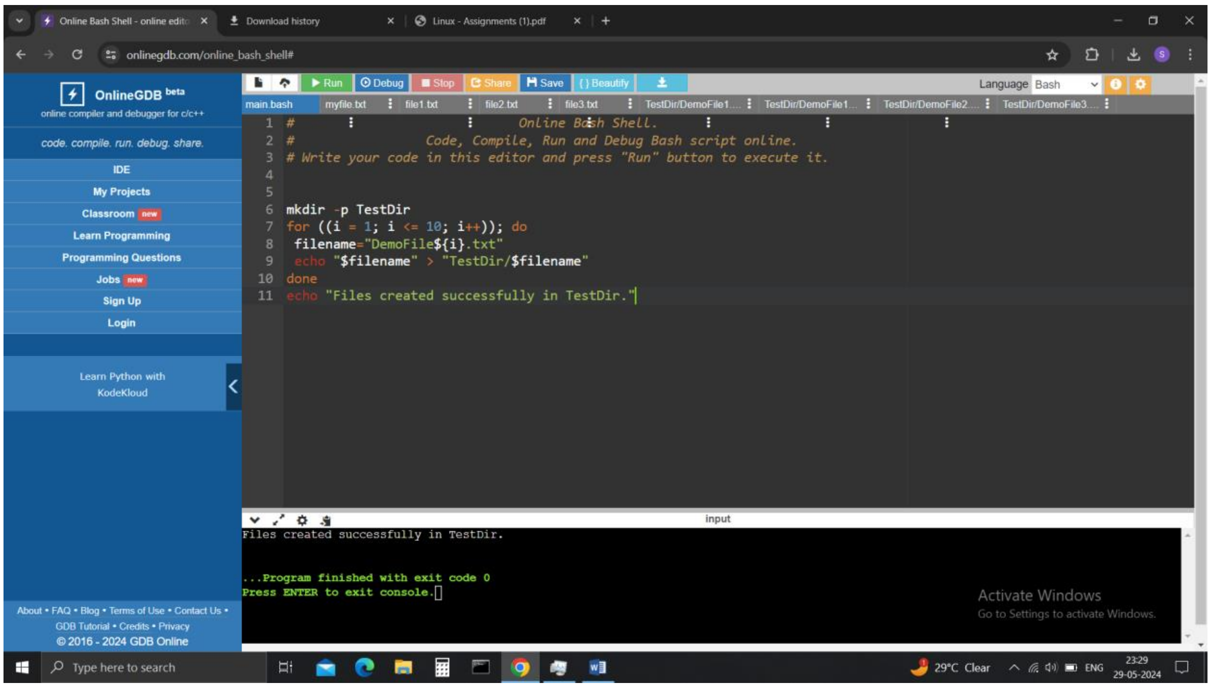
Task 3:Create a function that takes a filename as an argument and prints the number of lines in the file. Call this function from your script with different filenames.



```
#                           Online Bash Shell.
#                Code, Compile, Run and Debug Bash script online.
# Write your code in this editor and press "Run" button to execute it.


#!/bin/bassh

count_lines() {
  if [[ ! -f "$1" ]]; then
  echo "Error: File '$1' not found."
return 1
fi
num_lines=$(wc -l < "$1")
echo "File '$1' has $num_lines lines."
}



filenames=("file1.txt" "file2.txt" "file3.txt")
for filename in "${filenames[@]}"; do
 count_lines "$filename"
done
```

```
File 'file1.txt' has 1 lines.
File 'file2.txt' has 2 lines.
File 'file3.txt' has 3 lines.


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

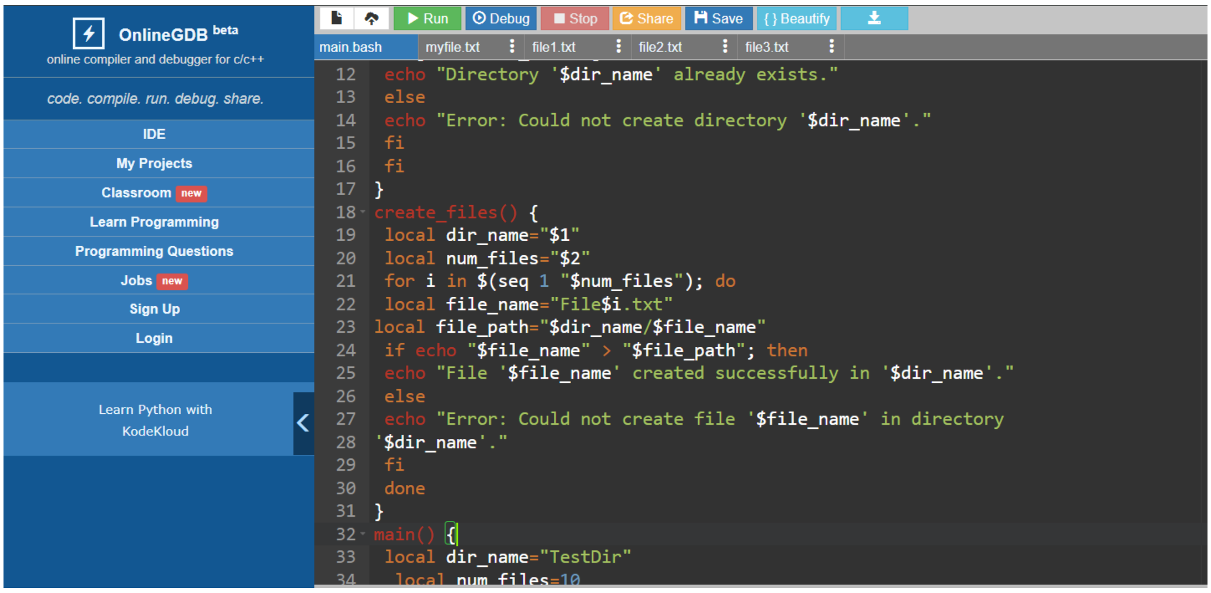
Task 4: Write a script that creates a directory named TestDir and inside it, creates ten files named File1.txt, File2.txt, ... File10.txt. Each file should contain its filename as its content (e.g., File1.txt contains "File1.txt").



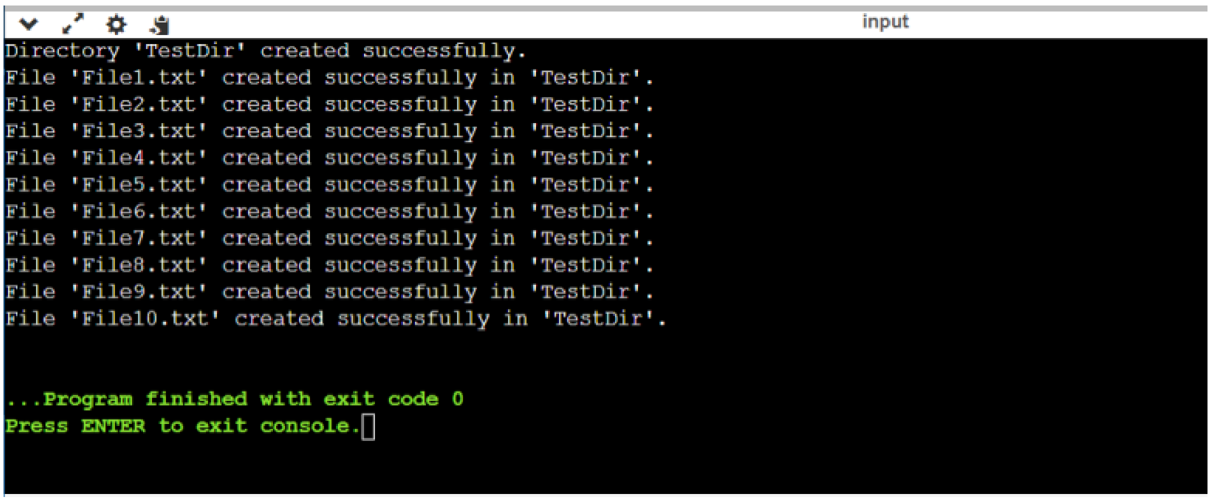
```
#                           Online Bash Shell.
#                Code, Compile, Run and Debug Bash script online.
# Write your code in this editor and press "Run" button to execute it.


mkdir -p TestDir
for ((i = 1; i <= 10; i++)); do
 filename="DemoFile${i}.txt"
 echo "$filename" > "TestDir/$filename"
done
echo "Files created successfully in TestDir."
```

```
Files created successfully in TestDir.


...Program finished with exit code 0
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.

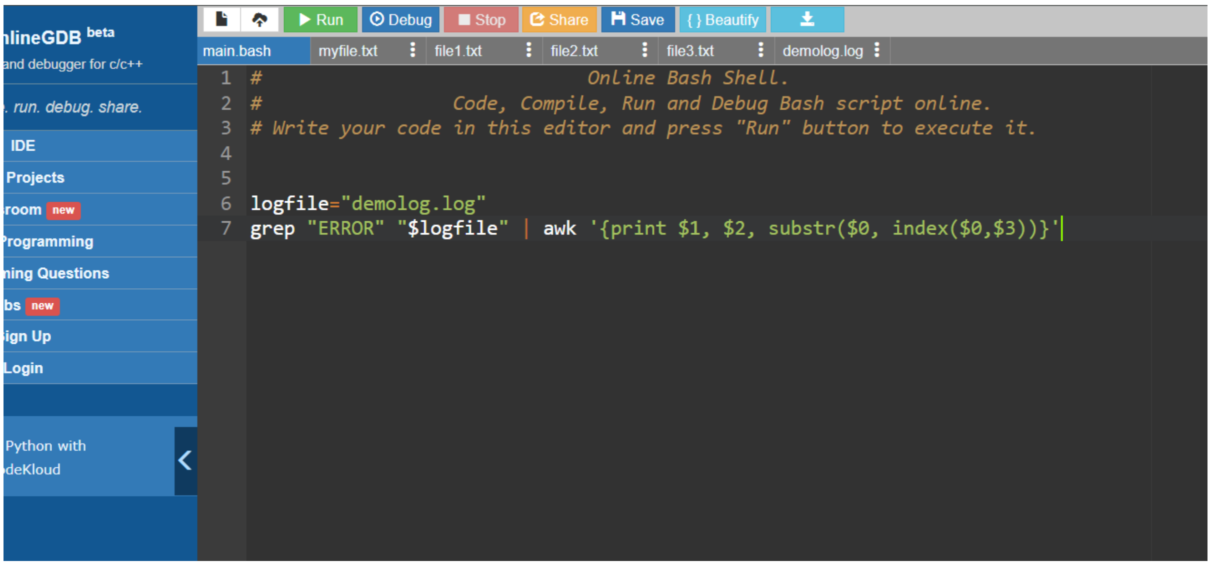


```
 echo "Directory '$dir_name' already exists."
 else
 echo "Error: Could not create directory '$dir_name'."
 fi
 fi
}
create_files() {
 local dir_name="$1"
 local num_files="$2"
 for i in $(seq 1 "$num_files"); do
 local file_name="File$i.txt"
local file_path="$dir_name/$file_name"
 if echo "$file_name" > "$file_path"; then
 echo "File '$file_name' created successfully in '$dir_name'."
 else
 echo "Error: Could not create file '$file_name' in directory
'$dir_name'."
 fi
 done
}
main() {
 local dir_name="TestDir"
  local num_files=10
```



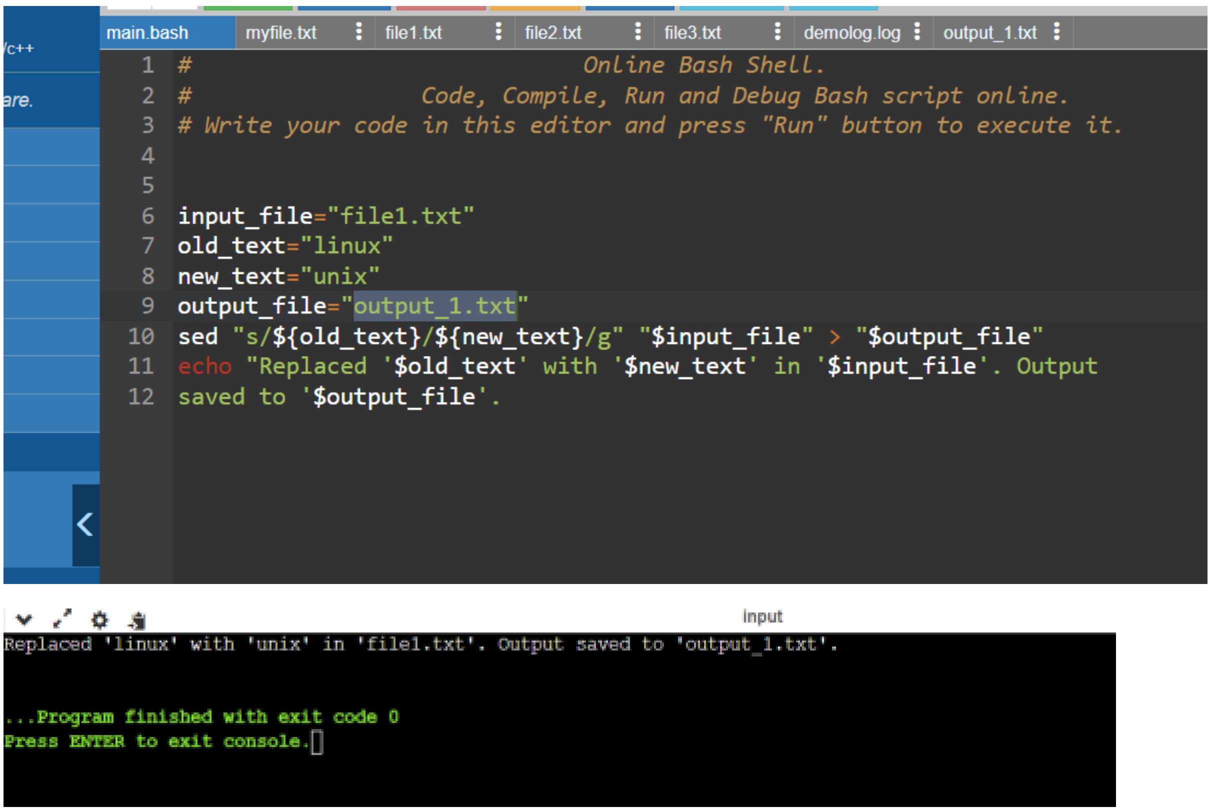
```
Directory 'TestDir' created successfully.
File 'File1.txt' created successfully in 'TestDir'.
File 'File2.txt' created successfully in 'TestDir'.
File 'File3.txt' created successfully in 'TestDir'.
File 'File4.txt' created successfully in 'TestDir'.
File 'File5.txt' created successfully in 'TestDir'.
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



```bash
#                            Online Bash Shell.
#                Code, Compile, Run and Debug Bash script online.
# Write your code in this editor and press "Run" button to execute it.


logfile="demolog.log"
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.



```bash
#                            Online Bash Shell.
#                Code, Compile, Run and Debug Bash script online.
# Write your code in this editor and press "Run" button to execute it.


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