

Assignment 1: Initialize a new Git repository in a directory of your choice. Add a simple text file to the repository and make the first commit.

```
MINGW64~/Users/Admin/Desktop/wip/practice
$ git init
Reinitialized existing Git repository in C:/Users/Admin/Desktop/wip/.git/

Admin@DESKTOP-QAR2QDJ MINGW64 ~/Desktop/wip (master)
$ config --global user.name "Vaishjagtap20"
bash: config: command not found

Admin@DESKTOP-QAR2QDJ MINGW64 ~/Desktop/wip (master)
$ config --global user.email "Vaishuj203@gmail.com"
bash: config: command not found

Admin@DESKTOP-QAR2QDJ MINGW64 ~/Desktop/wip (master)
$ config --global user.email Vaishuj203@gmail.com
bash: config: command not found

Admin@DESKTOP-QAR2QDJ MINGW64 ~/Desktop/wip (master)
$ git config --global user.name "Vaishjagtap20"

Admin@DESKTOP-QAR2QDJ MINGW64 ~/Desktop/wip (master)
$ git config --global user.email Vaishuj203@gmail.com

Admin@DESKTOP-QAR2QDJ MINGW64 ~/Desktop/wip (master)
$ cd practice
bash: cd: practice: No such file or directory

Admin@DESKTOP-QAR2QDJ MINGW64 ~/Desktop/wip (master)
$ git clone https://github.com/Vaishjagtap20/practice.git
Cloning into 'practice'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.

Admin@DESKTOP-QAR2QDJ MINGW64 ~/Desktop/wip (master)
$ cd practice

Admin@DESKTOP-QAR2QDJ MINGW64 ~/Desktop/wip/practice (main)
$ touch myFile.txt

Admin@DESKTOP-QAR2QDJ MINGW64 ~/Desktop/wip/practice (main)
$ git add .

Admin@DESKTOP-QAR2QDJ MINGW64 ~/Desktop/wip/practice (main)
$ git commit -m "Initial commit"
[main 5704db4] Initial commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 myFile.txt

Admin@DESKTOP-QAR2QDJ MINGW64 ~/Desktop/wip/practice (main)
$
```

Assignment 2: Branch Creation and Switching Create a new branch named 'feature' and switch to it. Make changes in the 'feature' branch and commit them.

```
Admin@DESKTOP-QAR2QDJ MINGW64 ~/Desktop/wip/practice (main)
$ git checkout -b feature
Switched to a new branch 'feature'

Admin@DESKTOP-QAR2QDJ MINGW64 ~/Desktop/wip/practice (feature)
$ git status
On branch feature
nothing to commit, working tree clean

Admin@DESKTOP-QAR2QDJ MINGW64 ~/Desktop/wip/practice (feature)
$ git add .

Admin@DESKTOP-QAR2QDJ MINGW64 ~/Desktop/wip/practice (feature)
$ git commit -m "branch create"
On branch feature
nothing to commit, working tree clean

Admin@DESKTOP-QAR2QDJ MINGW64 ~/Desktop/wip/practice (feature)
$
```

Assignment 3: Feature Branches and Hotfixes Create a 'hotfix' branch to fix an issue in the main code. Merge the 'hotfix' branch into 'main' ensuring that the issue is resolved.

```
MINGW64/c/Users/Admin/Desktop/wip/practice
AdminBDESKTOP-QAR2QD3 MINGW64 ~/Desktop/wip/practice (feature)
$ git checkout -b hotfix
Switched to a new branch 'hotfix'

AdminBDESKTOP-QAR2QD3 MINGW64 ~/Desktop/wip/practice (hotfix)
$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

AdminBDESKTOP-QAR2QD3 MINGW64 ~/Desktop/wip/practice (main)
$ git merge hotfix
Already up to date.

AdminBDESKTOP-QAR2QD3 MINGW64 ~/Desktop/wip/practice (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 282 bytes | 141.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Vaishjagtap20/practice.git
 bdf1696..5704db4  main -> main

AdminBDESKTOP-QAR2QD3 MINGW64 ~/Desktop/wip/practice (main)
$ git checkout feature
Switched to branch 'feature'

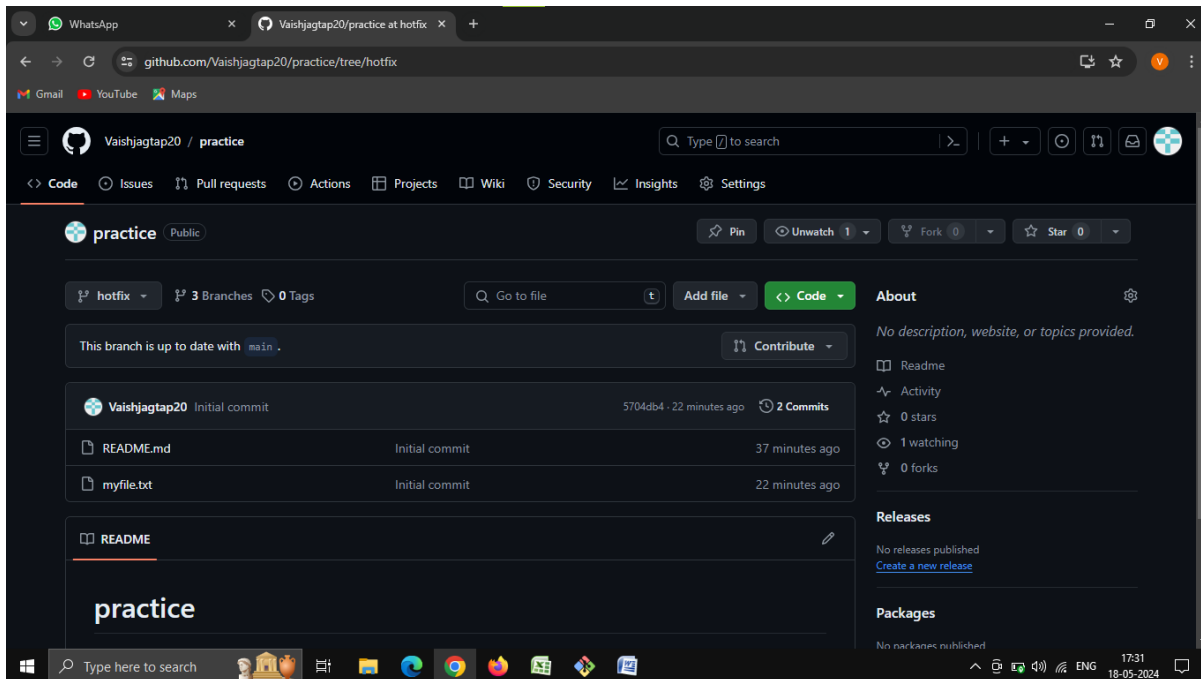
AdminBDESKTOP-QAR2QD3 MINGW64 ~/Desktop/wip/practice (feature)
$ git push
fatal: The current branch feature has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin feature

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetUpRemote' in 'git help config'.

AdminBDESKTOP-QAR2QD3 MINGW64 ~/Desktop/wip/practice (feature)
$ git push --set-upstream origin feature
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature' on GitHub by visiting:
remote:   https://github.com/Vaishjagtap20/practice/pull/new/feature
remote:
To https://github.com/Vaishjagtap20/practice.git
 * [new branch]   feature -> feature
```





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"

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

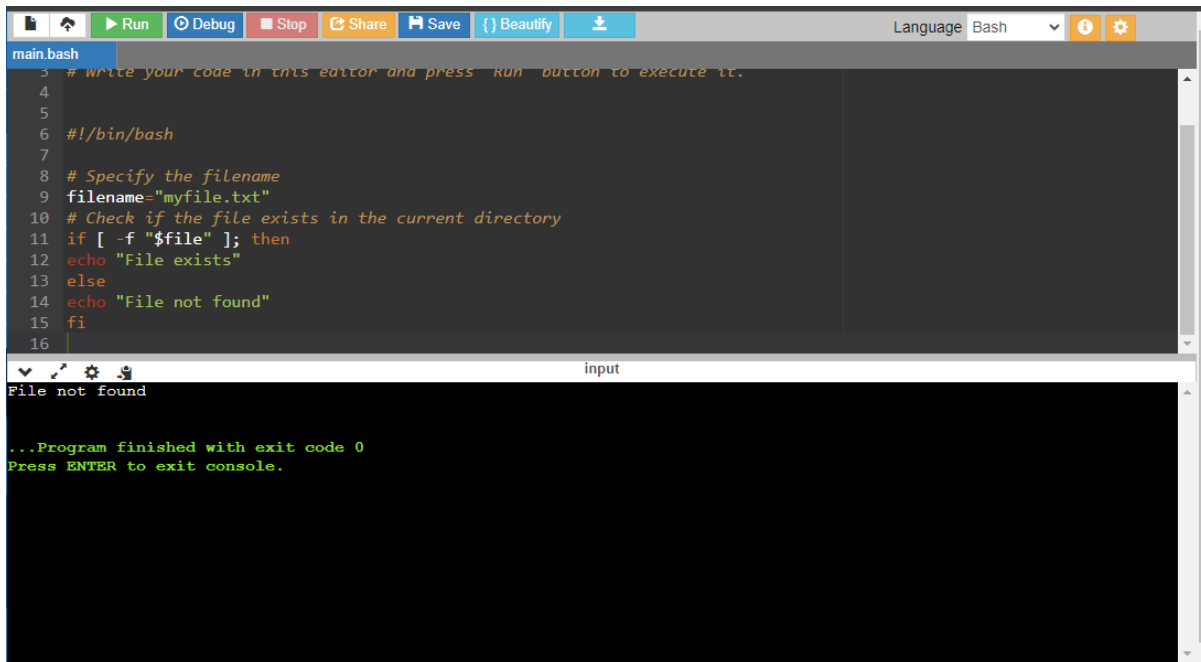
```
if [ -f "myfile.txt" ]; then
```

```
    echo "File exists"
```

```
else
```

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

```
fi
```



```
main.bash
3 # Write your code in this editor and press "Run" button to execute it.
4
5
6 #!/bin/bash
7
8 # Specify the filename
9 filename="myfile.txt"
10 # Check if the file exists in the current directory
11 if [ -f "$file" ]; then
12     echo "File exists"
13 else
14     echo "File not found"
15 fi
16
```

input

File not found

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

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 -p "Enter a number (0 to quit): " number
```

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

```
        break
```

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

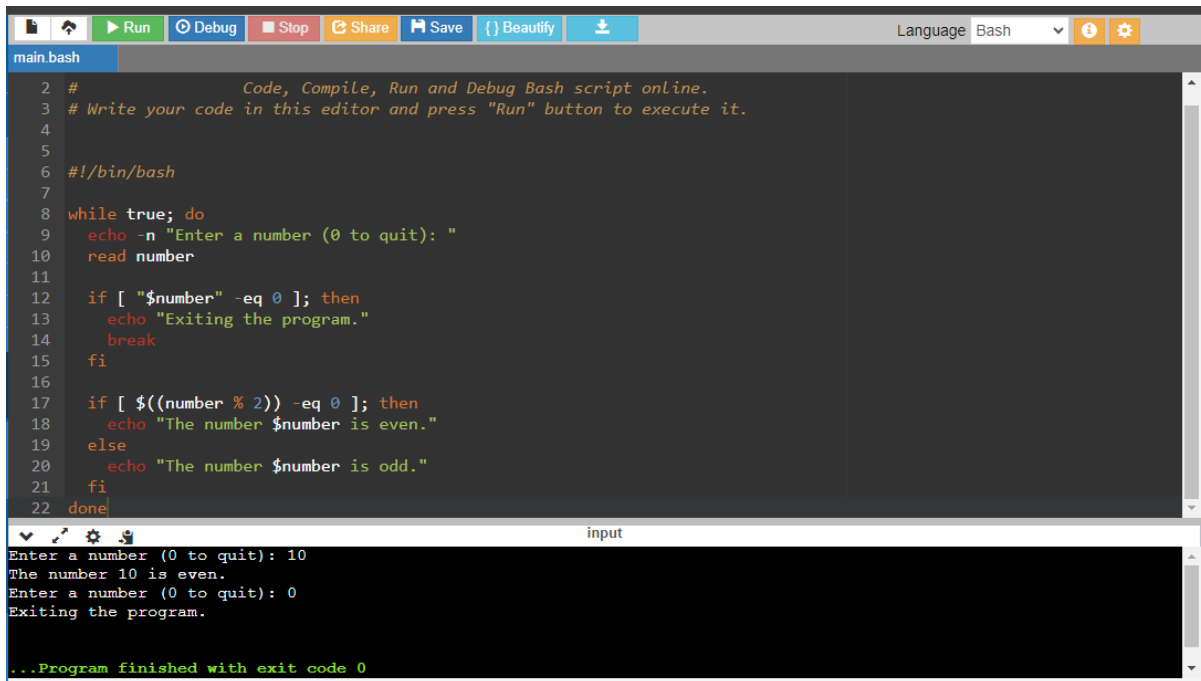
```
        echo "The number $number is even"
```

```
    else
```

```
        echo "The number $number is odd"
```

```
    fi
```

done



```
main.bash
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
8 while true; do
9     echo -n "Enter a number (0 to quit): "
10    read number
11
12    if [ "$number" -eq 0 ]; then
13        echo "Exiting the program."
14        break
15    fi
16
17    if [ $((number % 2)) -eq 0 ]; then
18        echo "The number $number is even."
19    else
20        echo "The number $number is odd."
21    fi
22 done
```

input

```
Enter a number (0 to quit): 10
The number 10 is even.
Enter a number (0 to quit): 0
Exiting the program.

...Program finished with exit code 0
```

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
```

```
        echo "Error: File '$1' not found."
```

```
        return 1
```

```
    fi
```

```
    num_lines=$(wc -l < "$1")
```

```
    echo "File '$1' has $num_lines lines."
```

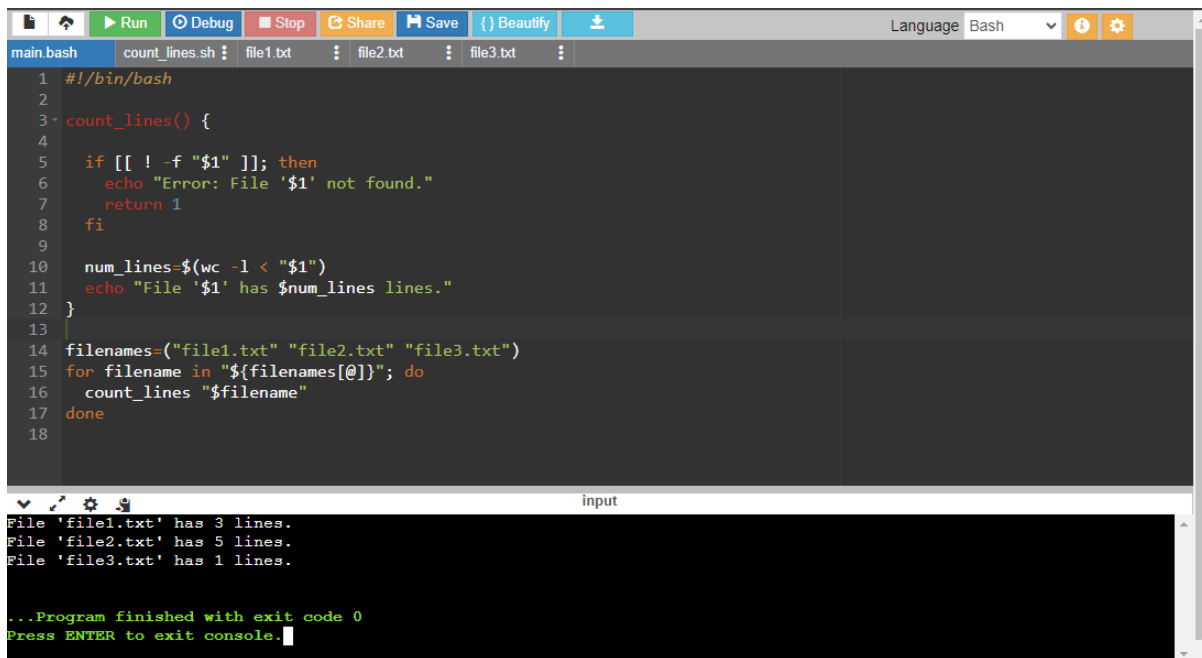
```
}
```

```
filenames=("file1.txt" "file2.txt" "file3.txt")
```

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

```
    count_lines "$filename"
```

```
done
```

A screenshot of a code editor window. The top bar shows tabs for 'main bash', 'count\_lines.sh', 'file1.txt', 'file2.txt', and 'file3.txt'. The 'count\_lines.sh' tab is active, displaying a Bash script. The script defines a function 'count\_lines' that checks if a file exists and prints its line count. It then iterates over an array of filenames ('file1.txt', 'file2.txt', 'file3.txt') and calls the function for each. The bottom panel shows the script's output: 'File 'file1.txt' has 3 lines.', 'File 'file2.txt' has 5 lines.', and 'File 'file3.txt' has 1 lines.'. It also shows the program finished with exit code 0 and a prompt to press ENTER to exit the console.

```
1 #!/bin/bash
2
3 count_lines() {
4     if [[ ! -f "$1" ]]; then
5         echo "Error: File '$1' not found."
6         return 1
7     fi
8
9     num_lines=$(wc -l < "$1")
10    echo "File '$1' has $num_lines lines."
11 }
12
13 filenames=("file1.txt" "file2.txt" "file3.txt")
14 for filename in "${filenames[@]}"; do
15     count_lines "$filename"
16 done
17
18
```

File 'file1.txt' has 3 lines.  
File 'file2.txt' has 5 lines.  
File 'file3.txt' has 1 lines.  
...Program finished with exit code 0  
Press ENTER to exit console.

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 -p TestDir
```

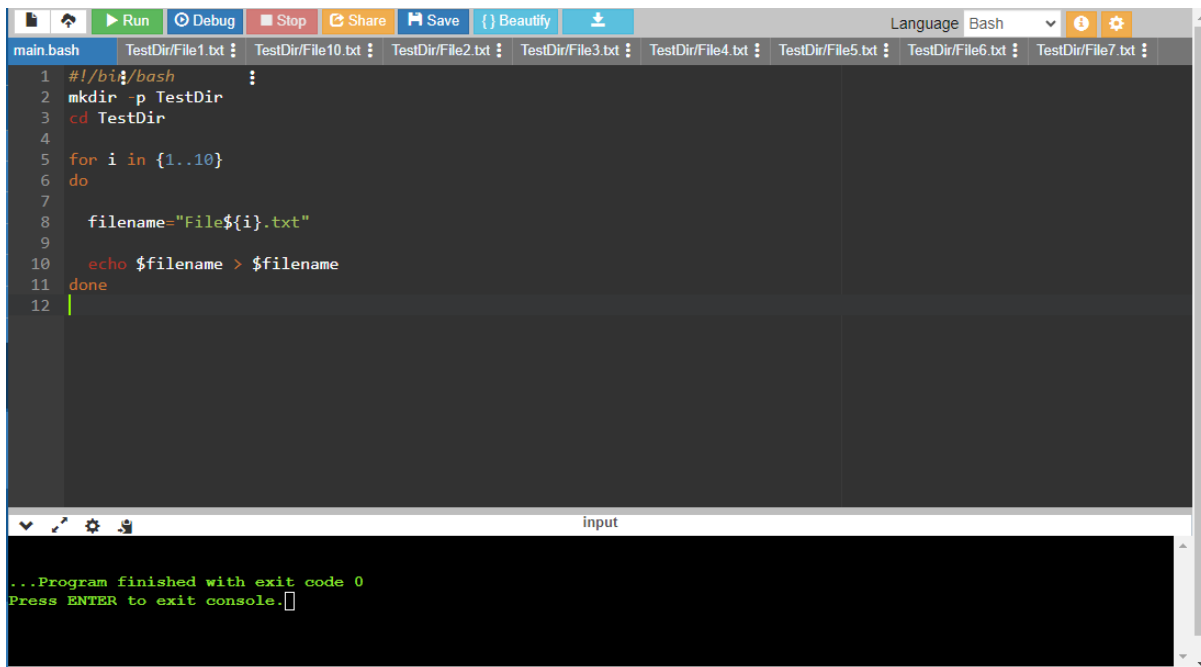
```
cd TestDir
```

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

    filename="File${i}.txt"

    echo $filename > $filename

done
```



The screenshot shows a code editor with a dark theme. The top toolbar includes buttons for Run, Debug, Stop, Share, Save, Beautify, and a download icon. The language is set to Bash. The editor displays a script with line numbers 1 through 12. The script creates a directory, changes to it, and runs a loop to create 10 files. Below the editor, an 'input' console shows the program finishing successfully with exit code 0.

```
1  #!/bin/bash
2  mkdir -p TestDir
3  cd TestDir
4
5  for i in {1..10}
6  do
7
8      filename="File${i}.txt"
9
10     echo $filename > $filename
11 done
12
```

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

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

```
#!/bin/bash

if [[ $1 == "debug" ]]; then
    debug_mode=true
else
```



```
    debug_mode=false
fi
debuglog()
{
    if [ "$debug_mode" = true ]; then
        echo "DEBUG: $1"
    fi
}

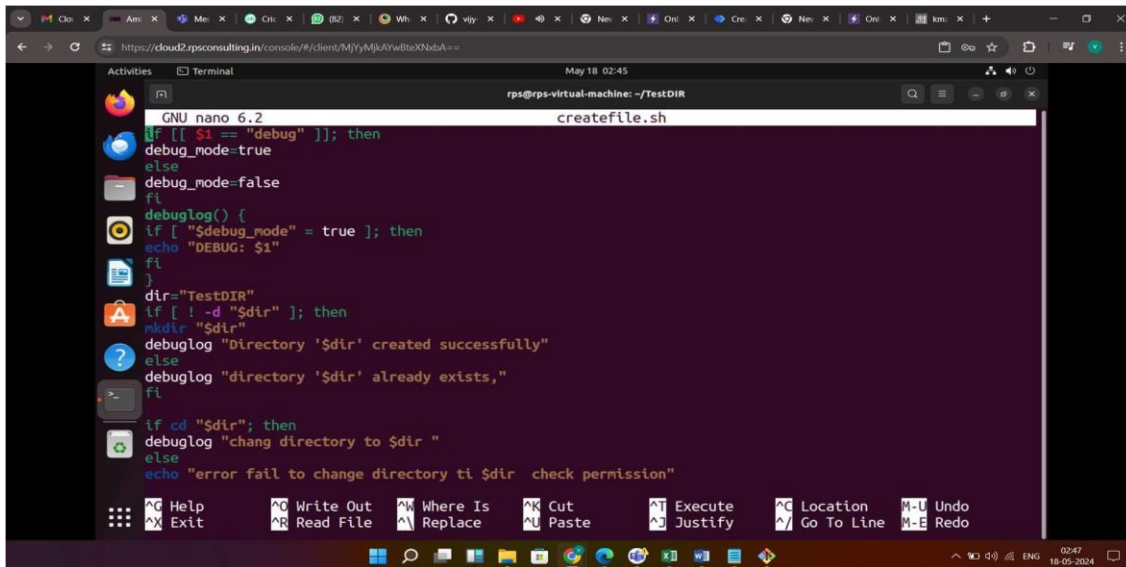
dir="TestDir"
if [ ! -d "$dir " ]; then
    mkdir "$dir "
    debuglog "Directory '$dir created successfully."
else
    debuglog "Directory '$dir already exists, skipping creation."
fi
if cd "$dir "; then
    debuglog "Changed directory to $dir."
else
    echo "Error: Failed to change directory to $dir . Check
permissions."
    exit 1
fi
for i in {1..10}; do
    f="File$i.txt"
    if echo "$f" > "$f"; then
```

debuglog "Created and wrote to \$f."

else

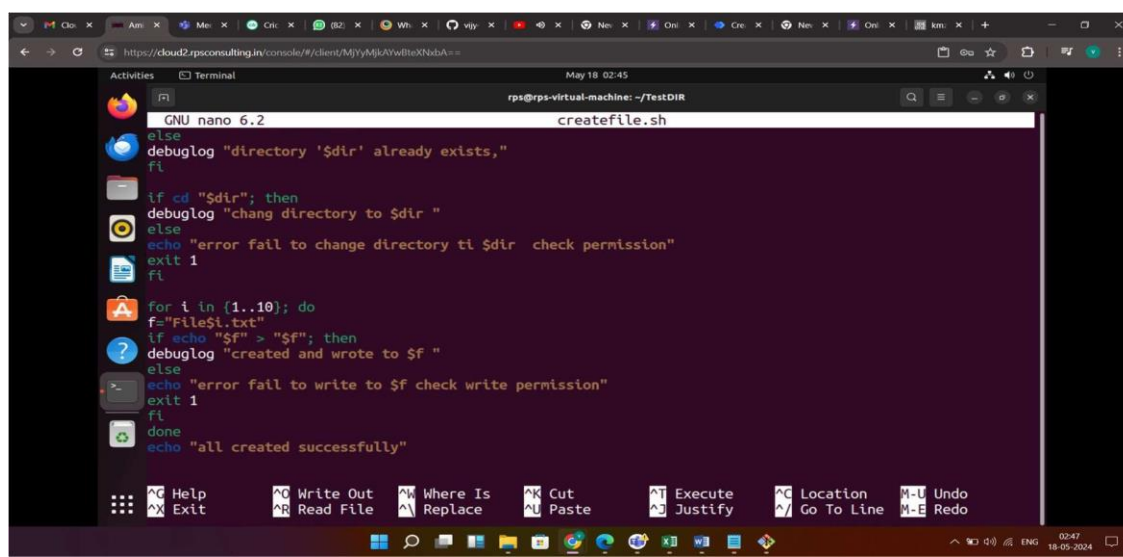
echo "Error: Failed to write to

\$f." exit 1



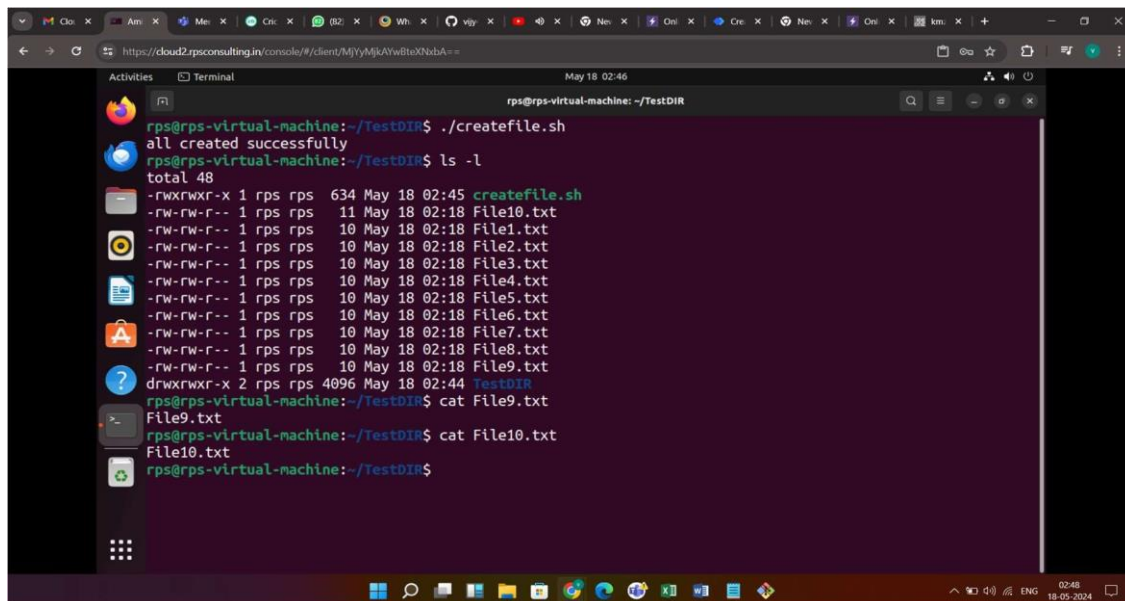
```
GNU nano 6.2 createfile.sh
if [[ $1 == "debug" ]]; then
  debug_mode=true
else
  debug_mode=false
fi
debuglog() {
  if [ "$debug_mode" = true ]; then
    echo "DEBUG: $1"
  fi
}
dir="TestDIR"
if [ ! -d "$dir" ]; then
  mkdir "$dir"
  debuglog "Directory '$dir' created successfully"
else
  debuglog "directory '$dir' already exists,"
fi

if cd "$dir"; then
  debuglog "chang directory to $dir "
else
  echo "error fail to change directory ti $dir  check permission"
```



```

  debuglog "directory '$dir' already exists,"
fi
if cd "$dir"; then
  debuglog "chang directory to $dir "
else
  echo "error fail to change directory ti $dir  check permission"
  exit 1
fi
for i in {1..10}; do
  f="File$i.txt"
  if echo "$f" > "$f"; then
    debuglog "created and wrote to $f "
  else
    echo "error fail to write to $f check write permission"
    exit 1
  fi
done
echo "all created successfully"
```

A screenshot of a terminal window titled "rps@rps-virtual-machine: ~/TestDIR". The terminal shows the execution of a script `./createfile.sh` which reports "all created successfully". This is followed by the command `ls -l`, which lists several files: `createfile.sh`, `File10.txt`, `File1.txt`, `File2.txt`, `File3.txt`, `File4.txt`, `File5.txt`, `File6.txt`, `File7.txt`, `File8.txt`, `File9.txt`, and `TestDIR`. The permissions and ownership details are visible for each file. The terminal also shows the output of `cat File9.txt` and `cat File10.txt`, both displaying "File9.txt" and "File10.txt" respectively. The terminal window is part of a desktop environment with various application icons visible on the left and bottom panels.

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

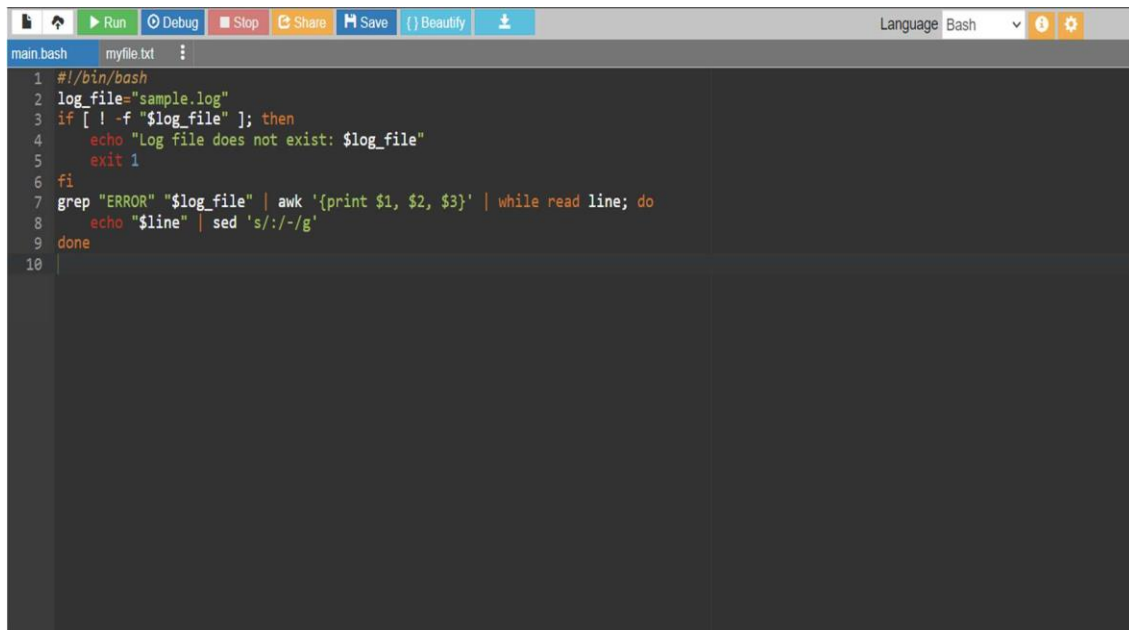
```
#!/bin/bash

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
```

A screenshot of a code editor window. The editor has a dark theme and a toolbar at the top with buttons for Run, Debug, Stop, Share, Save, and Beautify. The language is set to Bash. The script content is as follows:

```
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
```

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

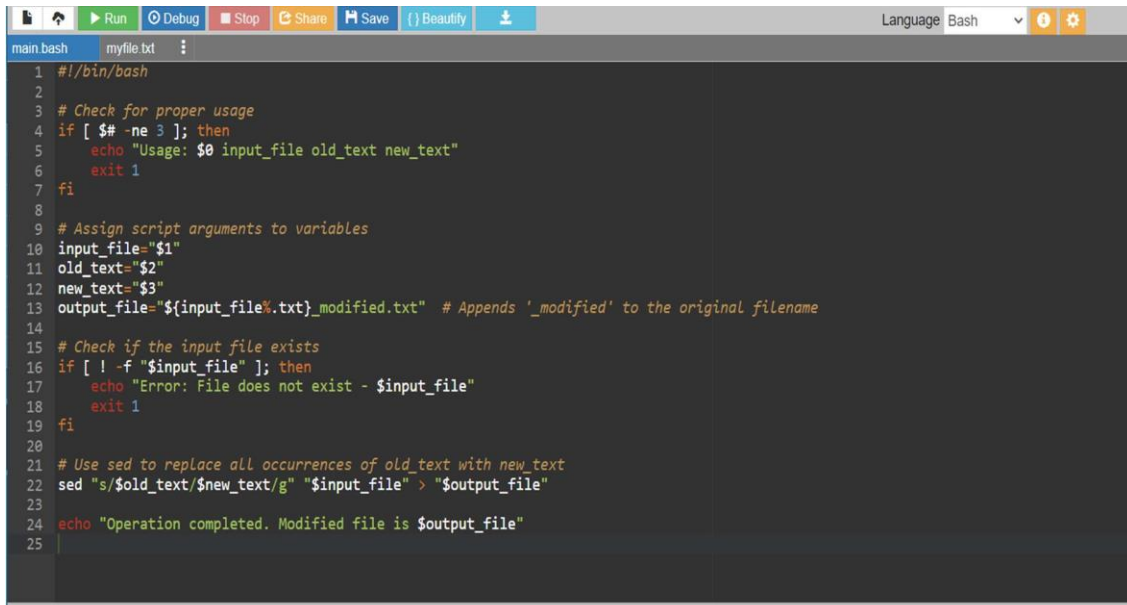
```
#!/bin/bash

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
d.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"
```

A screenshot of a code editor window. The top toolbar includes icons for Run, Debug, Stop, Share, Save, Beautify, and a download icon. The language is set to Bash. The editor shows a script named 'myfile.txt' with the following content:

```
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
```