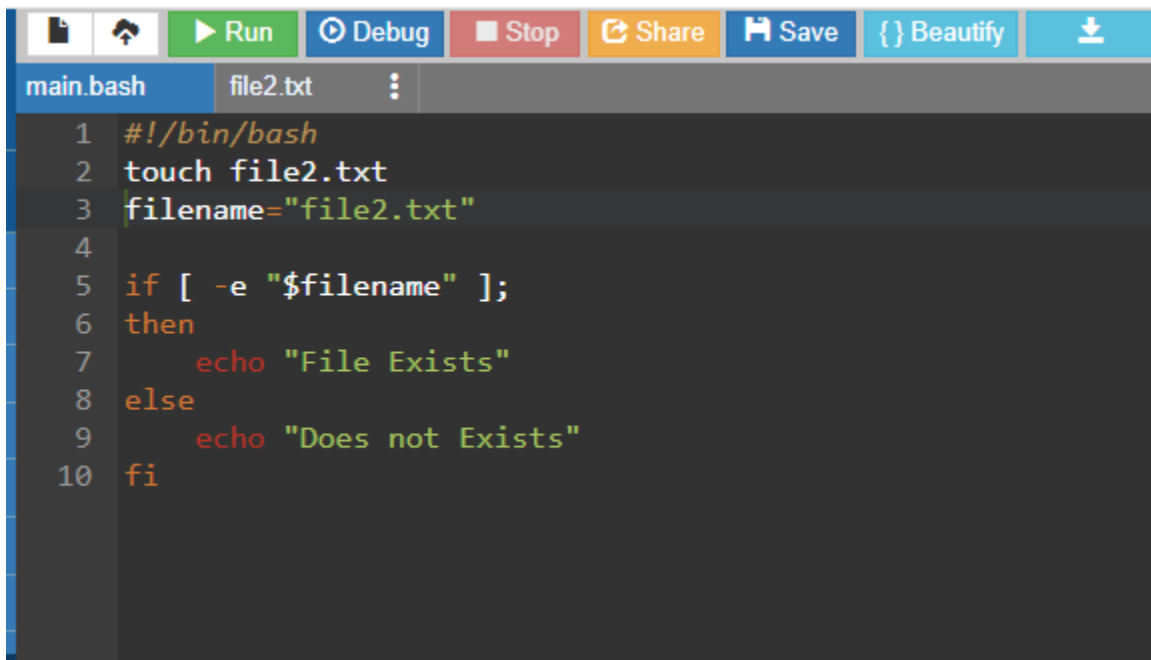


Assignment no 5

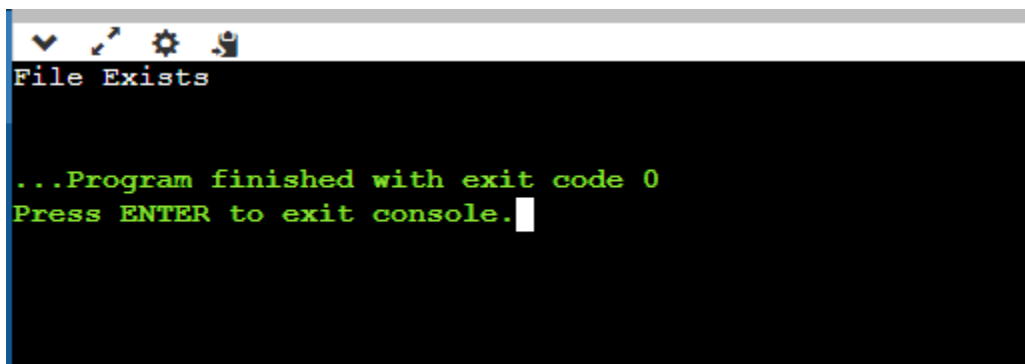
1] Ensure the script checks if a specific file (e.g., myfile.txt) exists in the current directory. If it exists, print "File exists", otherwise print "File not found".

Solution:-

A screenshot of a code editor interface. At the top, there is a toolbar with icons for file operations and buttons labeled 'Run', 'Debug', 'Stop', 'Share', 'Save', 'Beautify', and a download icon. Below the toolbar, the editor shows a file named 'file2.txt' with the following bash script:

```
1  #!/bin/bash
2  touch file2.txt
3  filename="file2.txt"
4
5  if [ -e "$filename" ];
6  then
7      echo "File Exists"
8  else
9      echo "Does not Exists"
10 fi
```

Output:-

A screenshot of a terminal window. The first line of output is 'File Exists'. Below this, there is a green message: '...Program finished with exit code 0'. At the bottom, it says 'Press ENTER to exit console.' with a white cursor character.

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.

Solution:-



The image shows a code editor window with a dark theme. At the top, there is a toolbar with icons for file operations and buttons for 'Run', 'Debug', 'Stop', 'Share', 'Save', 'Beautify', and a download icon. Below the toolbar, the editor has two tabs: 'main.bash' (active) and 'file2.txt'. The code in 'main.bash' is a bash script that runs in an infinite loop, prompting the user to enter numbers. It checks if the input is zero and if it is even or odd, providing feedback and exiting on zero.

```
1  #!/bin/bash
2
3  while true;
4  do
5
6      echo "Enter numbers "
7      read num
8
9      if [ "$num" -eq 0 ]
10     then
11         echo "zero is not allowed"
12         exit 1
13     fi
14
15     if [ $((num % 2)) == 0 ]
16     then
17         echo "$num is even"
18     else
19         echo "$num is odd"
20     fi
21 done
22
23
24
```

Output:-

```
Enter numbers
52
52 is even
Enter numbers
33
33 is odd
Enter numbers
0
zero is not allowed

...Program finished with exit code 1
Press ENTER to exit console.
```

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.

Solution:-

```
main.bash file2.txt file.txt file1.txt
1  #!/bin/bash
2  #touch file.txt file1.txt file2.txt
3  line_count(){
4  filename=$1
5  lines=$(wc -l < "$filename")
6  echo "Name of file is $filename and Number of lines in file are $lines"
7  }
8  line_count file.txt
9  line_count file1.txt
10 line_count file2.txt
```

Output:-

```
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, 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").

Solution:-

```
main.bash TestDir/File1.txt TestDir/File10.txt TestDir/File2.txt TestDir/File3.txt
1 | : : :
2
3 #!/bin/bash
4 mkdir TestDir
5 cd TestDir || exit
6
7 for ((i=1; i<=10; i++)); do
8
9     touch "File$i.txt"
10
11     echo "File$i.txt" > "File$i.txt"
12 done
13 echo "Files created successfully in TestDir."
```

Output:-

The screenshot shows a code editor with a dark theme. At the top, there are two tabs: 'main.bash' and 'TestDir/File1.txt'. The 'main.bash' tab is active, showing a script with two lines: '1 File1.txt' and '2'. Below the editor, there is a toolbar with buttons for 'Run', 'Debug', 'Stop', 'Share', 'Save', 'Beautify', and a download icon. To the right of the toolbar, there is a 'Language' dropdown menu set to 'Bash'. Below the toolbar, there are several tabs for files in 'TestDir': 'File1.txt', 'File10.txt', 'File2.txt', 'File3.txt', 'File4.txt', 'File5.txt', and 'File6.txt'. The 'File2.txt' tab is active, showing a script with two lines: '1' and '2'. Below the editor, there is a console window with the following output: 'Files created successfully in TestDir.', '...Program finished with exit code 0', and 'Press ENTER to exit console.'.

```
main.bash TestDir/File1.txt ⋮
1 File1.txt
2

Run Debug Stop Share Save {} Beautify ⬇ Language Bash ⓘ ⚙
main.bash TestDir/File1.txt ⋮ TestDir/File10.txt ⋮ TestDir/File2.txt ⋮ TestDir/File3.txt ⋮ TestDir/File4.txt ⋮ TestDir/File5.txt ⋮ TestDir/File6.txt ⋮
1 File2.txt ⋮ ⋮
2

Files created successfully in TestDir.
...Program finished with exit code 0
Press ENTER to exit console.␣
```

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.

Solution:-

Execute | Beautify | Share | Source Code | Help

```
1 #!/bin/bash
2
3
4 create_directory() {
5     local directory="$1"
6     local debug_mode="${2:-false}"
7
8     if [[ $debug_mode == true ]]; then
9         echo "Checking if directory '$directory' exists..."
10    fi
11
12    if [ ! -d "$directory" ]; then
13        mkdir -p "$directory"
14        if [[ $? -eq 0 ]]; then
15            if [[ $debug_mode == true ]]; then
16                echo "Directory '$directory' created successfully."
17            fi
18        else
19            echo "Error creating directory '$directory': Exit code ($?)"
20        fi
21    else
22        if [[ $debug_mode == true ]]; then
23            echo "Directory '$directory' already exists."
24        fi
25    fi
26 }
27
28
29 debug_mode=true
30 directory_to_create="my_new_directory"
31
32 create_directory "$directory_to_create" "$debug_mode"
```

```
26 }
27
28
29 debug_mode=true
30 directory_to_create="my_new_directory"
31
32 create_directory "$directory_to_create" "$debug_mode"
33
34 if [[ $? -eq 0 ]]; then
35     echo "Directory '$directory_to_create' created!"
36 fi
37
```

Output:-

```
Terminal
Checking if directory 'my_new_directory' exists...
Directory 'my_new_directory' created successfully.
Directory 'my_new_directory' created!
```

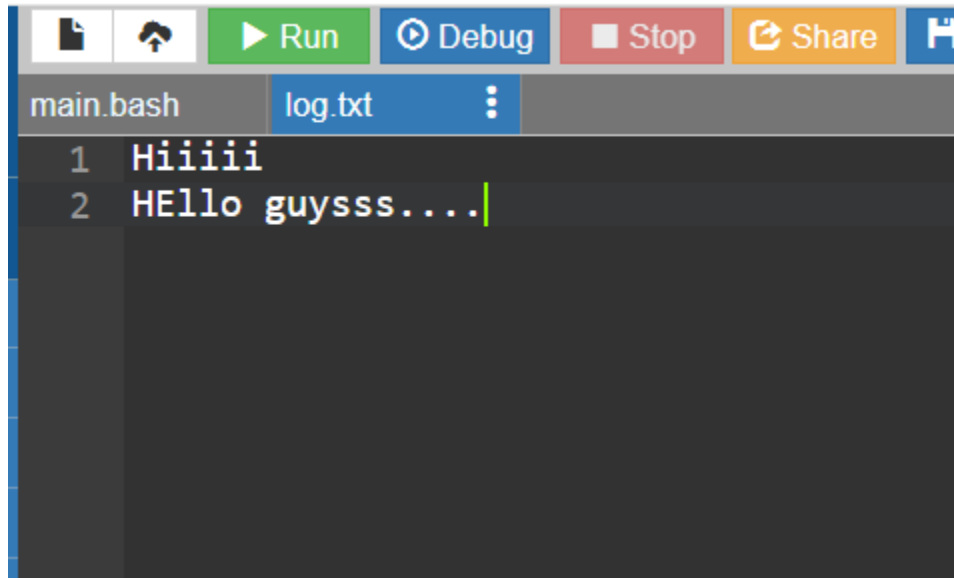
6] Given a sample log file, write a script using grep to extract all lines containing "ERROR". Use awk to print the date, time, and error message of each extracted line.

Data Processing with sed

Solution:-

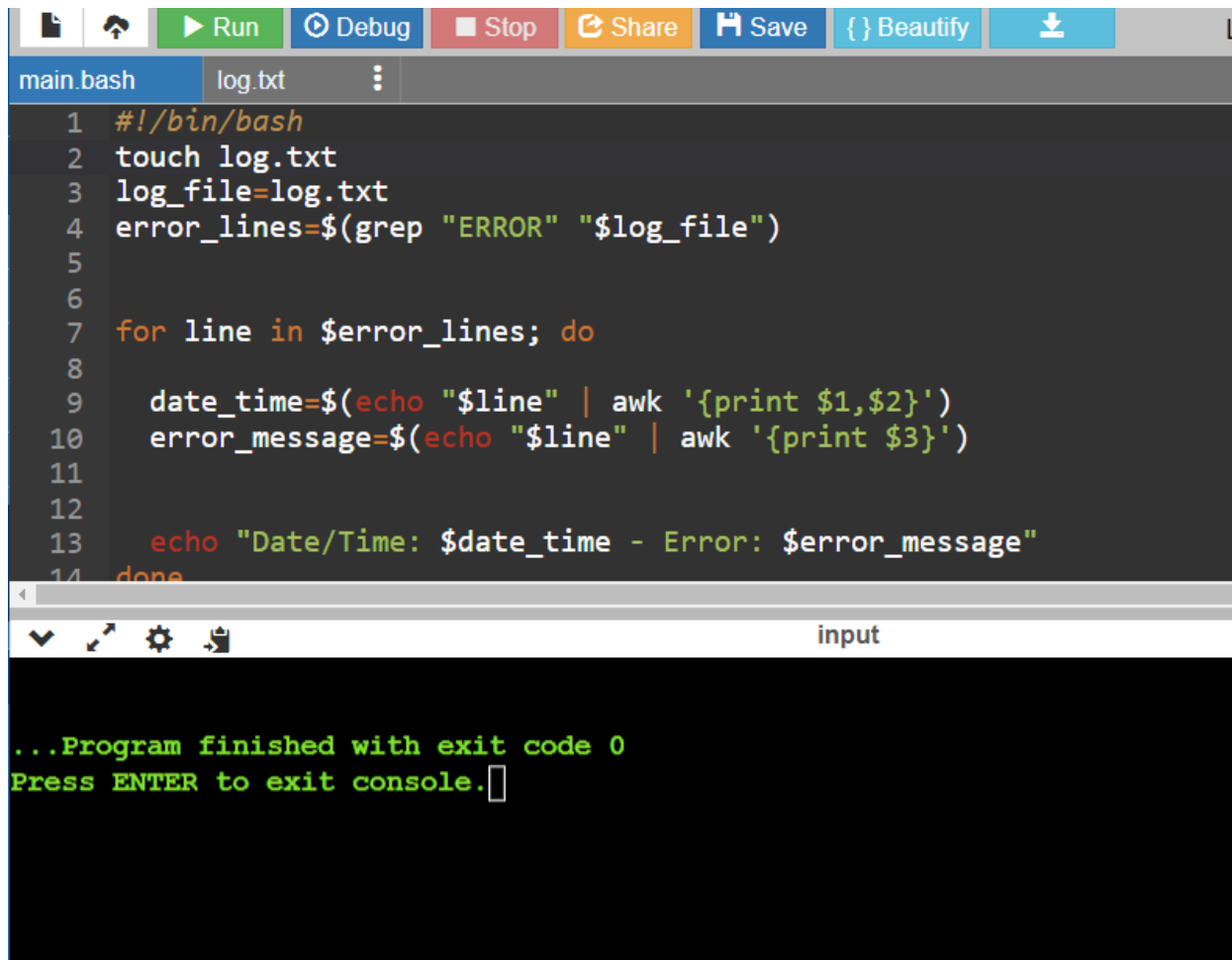
```
main.bash log.txt
1 #!/bin/bash
2 touch log.txt
3 log_file=log.txt
4 error_lines=$(grep "ERROR" "$log_file")
5
6
7 for line in $error_lines; do
8
9     date_time=$(echo "$line" | awk '{print $1,$2}')
10    error_message=$(echo "$line" | awk '{print $3}')
11
12
13    echo "Date/Time: $date_time - Error: $error_message"
14 done
```

Log.txt file-



```
main.bash log.txt
1 Hiiiii
2 HEllO guysss....|
```

Output:-



The image shows a code editor window with a dark theme. The top bar contains icons for file operations and buttons for Run, Debug, Stop, Share, Save, Beautify, and a download icon. Below the top bar, there are tabs for 'main.bash' and 'log.txt'. The main editor area displays a bash script with line numbers 1 through 14. The script creates a log file, searches for error lines, and formats them with a date and time. Below the script, there is a console window with a title bar containing icons for window management and a label 'input'. The console shows the message '...Program finished with exit code 0' and 'Press ENTER to exit console.' followed by a cursor.

```
1 #!/bin/bash
2 touch log.txt
3 log_file=log.txt
4 error_lines=$(grep "ERROR" "$log_file")
5
6
7 for line in $error_lines; do
8
9     date_time=$(echo "$line" | awk '{print $1,$2}')
10    error_message=$(echo "$line" | awk '{print $3}')
11
12
13    echo "Date/Time: $date_time - Error: $error_message"
14 done
```

input

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

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.

Solution:-

```
1  if [ $# -ne 2 ];
2  then
3      echo "Usage : $0 <input_file> <new_text>"
4      exit 1
5  fi
6
7  input_file="$1"
8  new_text="$2"
9
10 fi [ ! -f "$input_file" ];
11 then
12     echo "Error: File '$input_file' does not exist."
13 fi
14
15 output_file="${input_file%.txt}.replaced.txt"
16
17 sed -i "s/old_text/$new_text/g" "$input_file"
18
19 if [ $? -eq 0 ]
20 then
21     echo "Successfully replaced 'old_text' with '$new_text' in '$input_file'."
22     echo "Output saved to: $output_file"
23 else
24     echo "Error: Failed to replace text in '$input_file'."
25 fi
```

Output:-

Terminal

Usage : main.sh <input_file> <new_text>