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".

Ans:

```
#!/bin/bash

file_name="myfile.txt"

if [ -e "$file_name" ]; then
   echo "File exists"
else
   echo "File not found"
fi
```

Save this script in a file (let's say `check_file.sh`), make it executable (`chmod +x check_file.sh`), and then run it. It will check if `myfile.txt` exists in the current directory and print "File exists" if it does, otherwise "File not found". You can replace `"myfile.txt"` with the name of the file you want to check.

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.

Ans:

```
#!/bin/bash


echo "Enter numbers. Enter '0' to exit."


while true; do
    read -p "Enter a number: " num
    if [ "$num" -eq 0 ]; then
        echo "Exiting..."
        break
    fi


    if [ "$((num % 2))" -eq 0 ]; then
        echo "$num is even."
    else
        echo "$num is odd."
    fi
```

```
done
```

Save this script in a file (let's say `odd_even_checker.sh`), make it executable (`chmod +x odd_even_checker.sh`), and then run it. It will continuously prompt the user to enter numbers. It then checks whether each number is odd or even and prints the result. When the user enters '0', the script exits.

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.

Ans:

```
#!/bin/bash

count_lines() {
    filename="$1"
    if [ -f "$filename" ]; then
        num_lines=$(wc -l < "$filename")
        echo "Number of lines in $filename: $num_lines"
    else
```

```
        echo "File '$filename' not found."
    fi
}

# Call the function with different filenames
count_lines "file1.txt"
count_lines "file2.txt"
```

Save this script in a file (let's say `line_counter.sh`), make it executable (`chmod +x line_counter.sh`), and then run it. Replace `"file1.txt"` and `"file2.txt"` with the filenames you want to check. The script will print the number of lines in each file. If a file is not found, it will print a message indicating so.

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").

Ans:

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

```
# Create the directory TestDir if it doesn't exist
mkdir -p TestDir

# Change to the TestDir directory
cd TestDir || exit

# Create ten files with the desired content
for ((i = 1; i <= 10; i++)); do
   echo "File$i.txt" > "File$i.txt"
done

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

Save this script in a file (let's say `create_files.sh`

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.

Ans:

```
#!/bin/bash

DEBUG=false

# Function to display debug messages
debug() {
  if [ "$DEBUG" = true ]; then
    echo "DEBUG: $1"
  fi
}

# Function to create files
create_files() {
  for ((i = 1; i <= 10; i++)); do
    debug "Creating File$i.txt"
    echo "File$i.txt" > "File$i.txt"
  done
```

```
}

# Main script
main() {
    # Check if TestDir already exists
    if [ -d "TestDir" ]; then
        echo "Error: TestDir already exists."
        exit 1
    fi

    # Attempt to create the directory
    mkdir -p TestDir
    if [ $? -ne 0 ]; then
        echo "Error: Failed to create directory TestDir."
        exit 1
    fi

    debug "TestDir created successfully."

    # Change to the TestDir directory
    cd TestDir || exit 1

    # Check if we have write permission in the directory
    if [ ! -w . ]; then
        echo "Error: Permission denied to create files in
TestDir."
        exit 1
    fi

    debug "Permission granted to create files in TestDir."

    # Call the function to create files
    create_files

    echo "Files created successfully in TestDir."
}

# Parse command-line arguments
while getopts ":d" opt; do
    case $opt in
    d)
      DEBUG=true
      debug "Debug mode enabled."
      ;;
    \?)
      echo "Invalid option: -$OPTARG" >&2
      exit 1
      ;;
    esac
done

# Call the main function
Main
```

To use the script:

1. Save it in a file (e.g., `create_files.sh`).
2. Make it executable (`chmod +x create_files.sh`).
3. Run it with optional `-d` flag to enable debug mode for additional information (`./create_files.sh -d`).

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

Ans:

```
#!/bin/bash


# Sample log file path

log_file="sample.log"


# Use grep to extract lines containing "ERROR", then use awk to print date, time, and error message

grep "ERROR" "$log_file" | awk '{print $1, $2, substr($0, index($0,$3))}'
```

Save this script in a file (let's say `extract_errors.sh`), make it executable (`chmod +x extract_errors.sh`), and then run it. Ensure that `sample.log` exists in the same directory or specify the correct path to your log file.

This script will extract lines containing "ERROR" from the log file and print the date, time, and error message of each extracted line. Adjust `log_file` variable to point to your actual log file.

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.

Ans:

```
#!/bin/bash

# Check if correct number of arguments are provided
if [ "$#" -ne 3 ]; then
    echo "Usage: $0 input_file output_file old_text"
    exit 1
fi

input_file="$1"
output_file="$2"
old_text="$3"
new_text="new_text"  # You can replace this with any desired new text

# Perform the replacement using sed and output to a new file
sed "s/$old_text/$new_text/g" "$input_file" > "$output_file"

echo "Replacement completed. Output saved to $output_file"
```

save this script in a file (e.g., **`replace_text.sh`**) and make it executable (**`chmod +x replace_text.sh`**). Then you can use it as follows:

```
./replace_text.sh input_file.txt output_file.txt old_text
```

Replace **`input_file.txt`**, **`output_file.txt`**, and **`old_text`** with your actual input file, desired output file, and text to be replaced, respectively.