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

output is :

```
rps@rps-virtual-machine:~/wipro$ vi checkfile.sh
rps@rps-virtual-machine:~/wipro$ chmod 777 checkfile.sh
rps@rps-virtual-machine:~/wipro$ ./checkfile.sh
file not found
rps@rps-virtual-machine:~/wipro$ touch myfile.text
rps@rps-virtual-machine:~/wipro$ ./checkfile.sh
file not found
rps@rps-virtual-machine:~/wipro$ cat checkfile.sh
filename="myfile.txt"
if [ -f "$filename" ]; then
        echo "file exists"
else
        echo "file not found "
fi
rps@rps-virtual-machine:~/wipro$ touch myfile.txt
rps@rps-virtual-machine:~/wipro$ ./checkfile.sh
file exists
rps@rps-virtual-machine:~/wipro$
```

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

read -p "Enter a number 0 to quit: " number
while  [[  $number -ne 0 ]]; do
  # Check if the number is even using modulo operator (%)
  If  (( number % 2 == 0 ));  then
    echo "$number is even"
  else
```

```
  echo "$number is odd"
 fi
 read -p "Enter a number (0 to quit): " number
done
echo "Exiting..."
```

output is :

```
rps@rps-virtual-machine:~/wipro$ vi evenodd.sh
rps@rps-virtual-machine:~/wipro$ chmod 777 evenodd.sh
rps@rps-virtual-machine:~/wipro$ ./evenodd.sh
enter a number 0 to quit:^[[F9
./evenodd.sh: line 3: $'[[\E[F9': command not found
Exiting....
rps@rps-virtual-machine:~/wipro$ ./evenodd.sh
enter a number 0 to quit:6
./evenodd.sh: line 3: [[6: command not found
Exiting....
rps@rps-virtual-machine:~/wipro$ vi evenodd.sh
rps@rps-virtual-machine:~/wipro$ ./evenodd.sh
enter a number 0 to quit:6
6 is even
entera number 0 to quit :7
7 is odd
entera number 0 to quit :0
Exiting....
rps@rps-virtual-machine:~/wipro$ cat evenodd.sh
#!/usr/bin/bash
read -p "enter a number 0 to quit:" number
while [[ $number -ne 0 ]]; do
        if (( number % 2 ==0 )); then
                echo "$number is even"
        else
                echo "$number is odd"
        fi
        read -p "entera number 0 to quit :" number
done
echo "Exiting...."
rps@rps-virtual-machine:~/wipro$
```

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
function count_lines {
 filename="$1"
 if [ -f "$filename" ]; then
  line_count=$(wc -l < "$filename")
  echo "$filename has $line_count lines."
 else
```

```
    echo "File '$filename' not found."
  fi
}
count_lines "myfile.txt"
count_lines "change_file.txt"
```

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
dir_name="TestDir"
num_files=10
if [ ! -d "$dir_name" ]; then
  mkdir -p "$dir_name" || { echo "Error creating directory '$dir_name'"; exit 1; }
fi
for i in $(seq 1 $num_files); do
  filename="File$i.txt"
  filepath="$dir_name/$filename"
  # Create the file and write content (redirect to avoid overwriting)
  echo "$filename" > "$filepath" || { echo "Error creating file '$filepath'"; exit 1; }
done
echo "Created directory '$dir_name' with $num_files files."
```

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
# Directory name
dir_name="TestDir"
# Number of files
num_files=10
# Enable debugging mode (set to true for additional info)
debug_mode=false
# Function to print debug message
function debug_print {
```

```
  if [[ "$debug_mode" == true ]]; then
    echo "[DEBUG] $1"
  fi
}
# Check if directory already exists (informative message)
if [ -d "$dir_name" ]; then
  echo "Directory '$dir_name' already exists. Skipping creation."
  exit 0
fi
# Create the directory (handle errors)
debug_print "Creating directory: $dir_name"
if ! mkdir -p "$dir_name"; then
  echo "Error: Insufficient permissions to create directory '$dir_name'."
  exit 1
fi
# Loop to create files with unique content
for i in $(seq 1 $num_files); do
  filename="File$i.txt"
  filepath="$dir_name/$filename"
  # Create the file and write content (redirect to avoid overwriting)
  debug_print "Creating file: $filepath"
  if ! echo "$filename" > "$filepath"; then
    echo "Error creating file '$filepath'."
    exit 1
  fi
done
echo "Created directory '$dir_name' with $num_files files.
```

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
# Define the log file path
log_file="sample.log"
# Use grep to extract lines containing "ERROR" and then use awk to print date, time, and error message
```

```
grep "ERROR" "$log_file" | awk '{print $1, $2, substr($0, index($0,$4))}'
```

Explanation:

- grep "ERROR" "$log_file": This command searches for lines containing "ERROR" in the specified log file.
- awk '{print $1, $2, substr($0, index($0,$4))}': This awk command is used to extract the date, time, and error message from each line containing "ERROR".
- $1 and $2 represent the first and second fields, which are the date and time.
- substr($0, index($0,$4)) extracts the error message starting from the fourth field (which is the timestamp). This ensures that even if the error message contains spaces, it is printed entirely.

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

# Get the old and new text from the user.
echo "Enter the old text to be replaced:"
read old_text
echo "Enter the new text:"
read new_text

# Get the input and output file names from the user.
echo "Enter the input file name:"
read input_file
echo "Enter the output file name:"
read output_file

# Replace all occurrences of "old_text" with "new_text" in the input file and output the result to the output file.
sed "s/$old_text/$new_text/g" $input_file > $output_file

# Print a message to the user.
echo "The replacement is complete. The output file is $output_file."
```