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.

```
If [ -f input.sh ]; then
echo “File Exist “
else
echo “File not found”
fi
```

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.

```
echo "Enter numbers. Enter '0' to exit."
while true; do
  read -p "Enter a number: " num
  if [[ "$num" -eq 0 ]]; then
    break
  fi
  if (( $num % 2 == 0 )); then
    echo "$num is even."
  else
    echo "$num is odd."
  fi
done
echo "Exiting."
```

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.

```
count_lines() {
   local filename=$1
   if [ -f "$filename" ]; then
      local line_count=$(wc -l < "$filename")
      echo "The file '$filename' has $line_count lines."
   else
      echo "The file '$filename' does not exist."
   fi
}
count_lines "file1.txt"
count_lines "file2.txt"
```

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.

```
mkdir -p TestDir
for i in {1..10}; do
  touch "TestDir/File$i.txt"
  echo "Files$i.txt" > "TestDir/File$i.txt"
done
echo "Files created successfully in TestDir."
```

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.

```
directory="TestDir"
if [ ! -d "$directory" ]; then
  mkdir "$directory"
  echo "Directory '$directory' created."
else
  echo "Directory '$directory' already exists."
fi
for i in {1..10}
do
  file_name="File$i.txt"
  file_path="$directory/$file_name"
  echo "$file_name" > "$file_path"
  echo "File '$file_name' created with content '$file_name'."
done
```

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

log_file="logfile.log"

error_lines=$(grep "ERROR" "$log_file")

echo "Date       Time       Error Message"

for line in $error_lines; do
  date=$(echo "$line" | awk '{print $1}')
  time=$(echo "$line" | awk '{print $2}')
  message=$(echo "$line" | awk '{print $3}')
  printf "%-10s %-10s %s\n" "$date" "$time" "$message"
done
```

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.

```
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="output.txt"

sed "s/$old_text/$new_text/g" "$input_file" > "$output"

echo "Replaced all occurrences of '$old_text' with '$new_text' in '$input_file'."

echo "Output saved to '$output'."
```