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"

```
# File name to check
file="maria.txt"



# Check if the file exists
if [ -e "$file" ]; then
  echo "File exists"
else
  echo "File not found"
fi
```

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.

```
# Function to check if a number is odd or even
check_odd_even() {
  if [ $(( $1 % 2 )) -eq 0 ]; then
    echo "$1 is even"
  else
    echo "$1 is odd"
  fi
}



# Loop to read numbers from the user
while true; do
  echo -n "Enter a number (0 to quit): "
  read number

  if [ "$number" -eq 0 ]; then
    echo "Exiting..."
    break
  fi

  check_odd_even "$number"
done
```

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.

```
# Function to count the number of lines in a file
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
}



# Call the function with different filenames
count_lines "file1.txt"
count_lines "file2.txt"
count_lines "file3.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").

```
# Create a directory named TestDirOld
mkdir -p TestDirOld



# Change to the TestDirOld directory
cd TestDirOld
```

```
# Loop to create 10 files
for i in {1..10}; do
  filename="File$i.txt"
  echo "$filename" > "$filename"
done



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

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.

```
# Set debugging mode (0 for off, 1 for on)
DEBUG=1



# Function to print debug messages
debug() {
  if [ $DEBUG -eq 1 ]; then
    echo "DEBUG: $1"
  fi
}



# Create a directory named TestDirNew
if mkdir -p TestDirNew 2>/dev/null; then
  debug "Directory 'TestDirNew' created or already exists."
else
  echo "Error: Unable to create directory 'TestDirNew'. Check permissions."
  exit 1
fi



# Change to the TestDirNew directory
if cd TestDirNew; then
```

```
  debug "Changed to directory 'TestDirNew'."
else
  echo "Error: Unable to change to directory 'TestDirNew'. Check permissions."
  exit 1
fi



# Loop to create 10 files
for i in {1..10}; do
  filename="File$i.txt"
  if echo "$filename" > "$filename" 2>/dev/null; then
    debug "File '$filename' created with content '$filename'."
  else
    echo "Error: Unable to create file '$filename'. Check permissions."
    exit 1
  fi
done



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

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

Use 'grep' to extract lines containing "ERROR".
Use 'awk' to print the date, time, and error message.
Optionally, use 'sed' for further processing if needed.

```
# Check if the log file is provided as an argument
if [ -z "$1" ]; then
  echo "Usage: $0 <logfile>"
  exit 1
fi
```

```
LOGFILE="$1"


# Extract lines containing "ERROR" and process with awk
grep "ERROR" "$LOGFILE" | awk '{print $1, $2, $3, $4, $5}'


#Run the script with your log file
./extract_errors.sh sample.log


#Given the example log file provided, the output would be
2024-05-24 14:33:45 ERROR An error occurred in the service
2024-05-24 14:36:05 ERROR Another error occurred


#This output shows the date, time, and error message for each line containing "ERROR".
#If additional text processing is needed, sed can be incorporated. For example, to remove
extra spaces in the error message:


#!/bin/bash


# Check if the log file is provided as an argument
if [ -z "$1" ]; then
  echo "Usage: $0 <logfile>"
  exit 1
fi


LOGFILE="$1"


# Extract lines containing "ERROR", process with awk, and further refine with sed
grep "ERROR" "$LOGFILE" | awk '{print $1, $2, $3, $4, substr($0, index($0,$5))}' | sed 's/
\+/ /g'
```

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.

```
# Check if the correct number of arguments is provided
if [ "$#" -ne 4 ]; then
    echo "Usage: $0 input_file old_text new_text output_file"
    exit 1
fi


# Assign input arguments to variables
INPUT_FILE=$1
OLD_TEXT=$2
NEW_TEXT=$3
OUTPUT_FILE=$4


# Use sed to replace all occurrences of old_text with new_text
sed "s/$OLD_TEXT/$NEW_TEXT/g" "$INPUT_FILE" > "$OUTPUT_FILE"


# Print a message indicating the operation is complete
echo "Replaced all occurrences of '$OLD_TEXT' with '$NEW_TEXT' in '$INPUT_FILE' and saved
```