

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".

Answer:

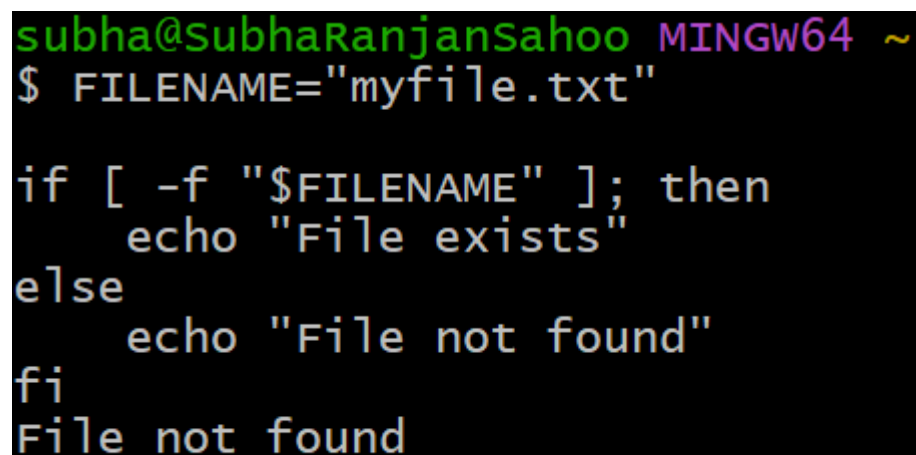
```
subha@SubhaRanjanSahoo MINGW64 ~
```

```
$ FILENAME="myfile.txt"

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

Output:

File not found



```
subha@SubhaRanjanSahoo MINGW64 ~
$ FILENAME="myfile.txt"

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

Assignment 2

Q. 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.

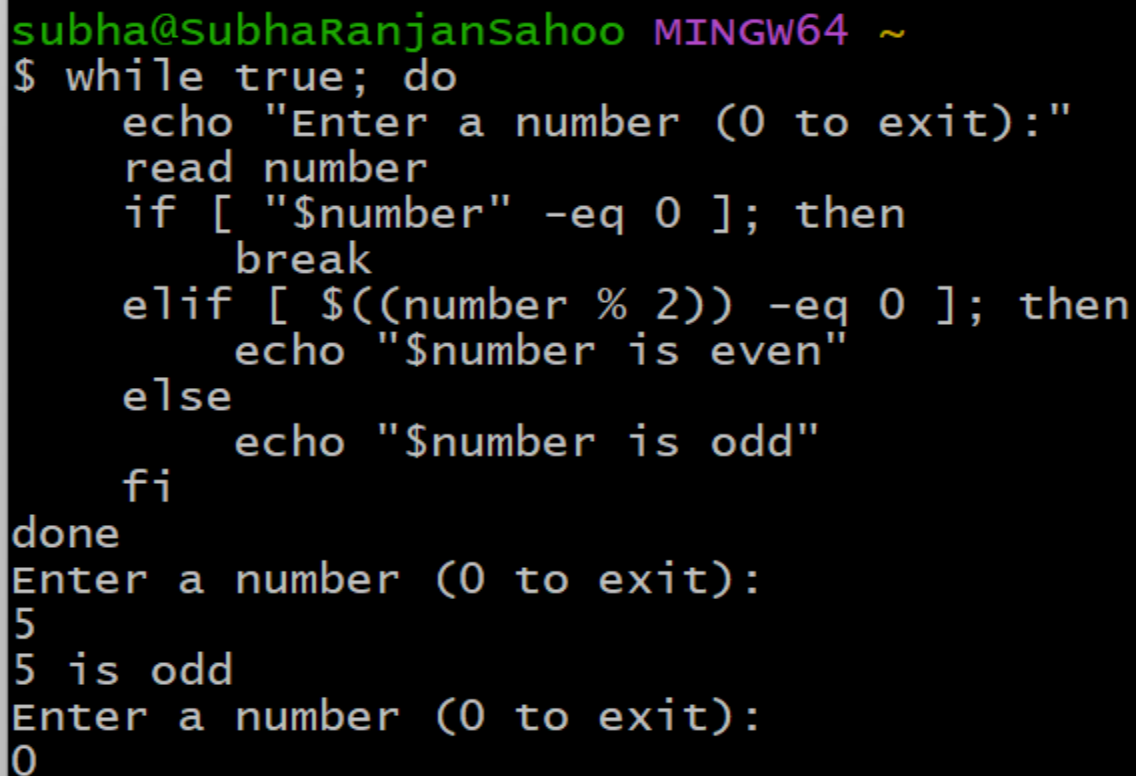
Answer:

```
subha@SubhaRanjanSahoo MINGW64 ~
```

```
$ while true; do
    echo "Enter a number (0 to exit):"
```

```
    read number
    if [ "$number" -eq 0 ]; then
        break
    elif [ $((number % 2)) -eq 0 ]; then
        echo "$number is even"
    else
        echo "$number is odd"
    fi
done
Enter a number (0 to exit):
5
5 is odd
Enter a number (0 to exit):
0
Output:
```

```
Enter a number (0 to exit):
5
5 is odd
Enter a number (0 to exit):
9
9 is odd
Enter a number (0 to exit):
4
4 is even
Enter a number (0 to exit):
0
```

A terminal window with a black background and white text. The prompt is 'subha@SubhaRanjanSahoo MINGW64 ~'. The script being executed is a while loop that prompts the user to enter a number (0 to exit). It checks if the number is 0 (break), even (echo "\$number is even"), or odd (echo "\$number is odd"). The user enters 5, 9, 4, and 0 in sequence.

```
subha@SubhaRanjanSahoo MINGW64 ~
$ while true; do
    echo "Enter a number (0 to exit):"
    read number
    if [ "$number" -eq 0 ]; then
        break
    elif [ $((number % 2)) -eq 0 ]; then
        echo "$number is even"
    else
        echo "$number is odd"
    fi
done
Enter a number (0 to exit):
5
5 is odd
Enter a number (0 to exit):
9
9 is odd
Enter a number (0 to exit):
4
4 is even
Enter a number (0 to exit):
0
```

Assignment 3

Q. 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.

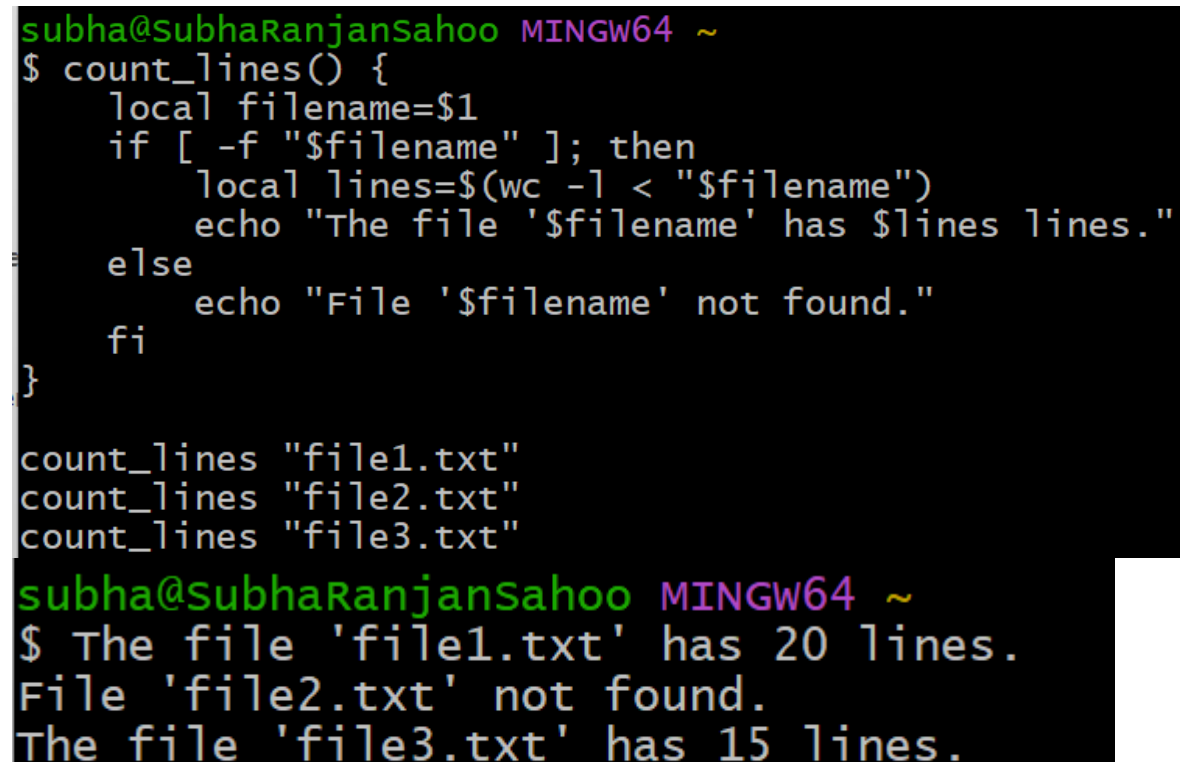
Answer:

```
subha@SubhaRanjanSahoo MINGW64 ~
$ count_lines() {
    local filename=$1
    if [ -f "$filename" ]; then
        local lines=$(wc -l < "$filename")
        echo "The file '$filename' has $lines lines."
    else
        echo "File '$filename' not found."
    fi
}

count_lines "file1.txt"
count_lines "file2.txt"
count_lines "file3.txt"
```

Output:

```
The file 'file1.txt' has 20 lines.
File 'file2.txt' not found.
The file 'file3.txt' has 15 lines.
```



```
subha@SubhaRanjanSahoo MINGW64 ~
$ count_lines() {
    local filename=$1
    if [ -f "$filename" ]; then
        local lines=$(wc -l < "$filename")
        echo "The file '$filename' has $lines lines."
    else
        echo "File '$filename' not found."
    fi
}

count_lines "file1.txt"
count_lines "file2.txt"
count_lines "file3.txt"

subha@SubhaRanjanSahoo MINGW64 ~
$ The file 'file1.txt' has 20 lines.
File 'file2.txt' not found.
The file 'file3.txt' has 15 lines.
```

Assignment 4

Q. 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").

Answer:

```
#!/bin/bash

DIRNAME="TestDir"

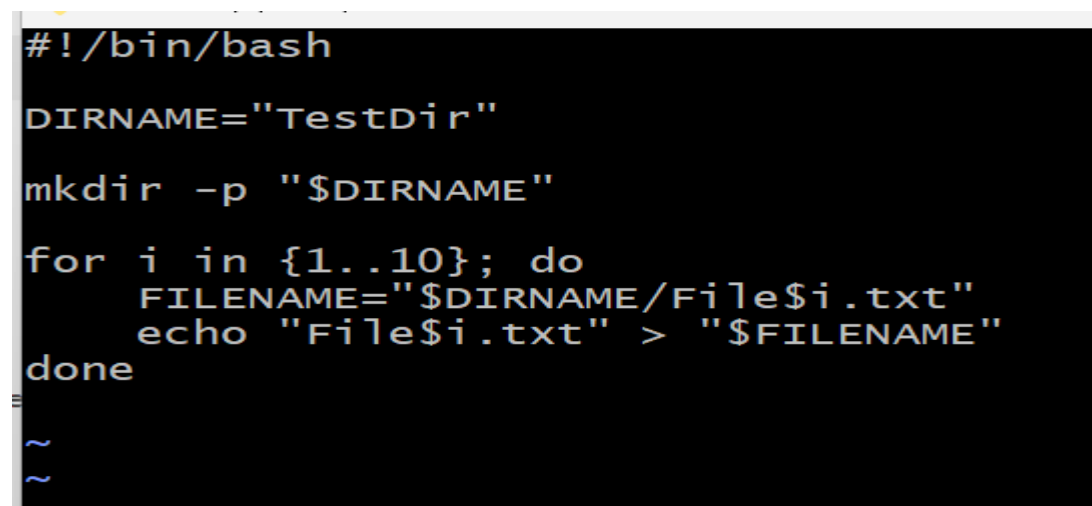
mkdir -p "$DIRNAME"

for i in {1..10}; do
    FILENAME="$DIRNAME/File$i.txt"
    echo "File$i.txt" > "$FILENAME"
done
```

Output:

```
subha@SubhaRanjanSahoo MINGW64 ~
$ ls TestDir/
File1.txt  File2.txt  File4.txt  File6.txt
File8.txt
File10.txt File3.txt  File5.txt  File7.txt
File9.txt

subha@SubhaRanjanSahoo MINGW64 ~
$ cat TestDir/File1.txt
File1.txt
```



```
#!/bin/bash

DIRNAME="TestDir"

mkdir -p "$DIRNAME"

for i in {1..10}; do
    FILENAME="$DIRNAME/File$i.txt"
    echo "File$i.txt" > "$FILENAME"
done
```

```

subha@SubhaRanjanSahoo MINGW64 ~
$ ls TestDir/
File1.txt  File2.txt  File4.txt  File6.txt  File8.txt
File10.txt File3.txt  File5.txt  File7.txt  File9.txt

subha@SubhaRanjanSahoo MINGW64 ~
$ cat TestDir/File1.txt
File1.txt

```

Assignment 5

Q. 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.

Answer:

```

#!/bin/bash

DEBUG=true
DIRNAME="TestDir"

if $DEBUG; then
    echo "Debugging mode is ON"
fi

if [ -d "$DIRNAME" ]; then
    echo "Directory '$DIRNAME' already exists."
else
    mkdir "$DIRNAME" || { echo "Failed to create
directory '$DIRNAME'"; exit 1; }
    if $DEBUG; then
        echo "Directory '$DIRNAME' created."
    fi
fi

for i in {1..10}; do
    FILENAME="$DIRNAME/File$i.txt"
    echo "File$i.txt" > "$FILENAME" || { echo "Failed
to create file '$FILENAME'"; exit 1; }
    if $DEBUG; then
        echo "Created file '$FILENAME' with content
'File$i.txt'"
    fi
done

```

Output:

```
subha@SubhaRanjanSahoo MINGW64 ~ with content
'File1.txt'
$ eated file 'TestDir/File2.txt' with content
'File2.txt'
Created file 'TestDir/File3.txt' with content
'File3.txt'
Created file 'TestDir/File4.txt' with content
'File4.txt'
Created file 'TestDir/File5.txt' with content
'File5.txt'
Created file 'TestDir/File6.txt' with content
'File6.txt'
Created file 'TestDir/File7.txt' with content
'File7.txt'
Created file 'TestDir/File8.txt' with content
'File8.txt'
Created file 'TestDir/File9.txt' with content
'File9.txt'
Created file 'TestDir/File10.txt' with content
'File10.txt'
```

```
#!/bin/bash
DEBUG=true
DIRNAME="TestDir"

if $DEBUG; then
    echo "Debugging mode is ON"
fi

if [ -d "$DIRNAME" ]; then
    echo "Directory '$DIRNAME' already exists."
else
    mkdir "$DIRNAME" || { echo "Failed to create directory '$DIRNAME'"; exit 1; }
    if $DEBUG; then
        echo "Directory '$DIRNAME' created."
    fi
fi

for i in {1..10}; do
    FILENAME="$DIRNAME/File$i.txt"
    echo "File$i.txt" > "$FILENAME" || { echo "Failed to create file '$FILENAME'"; exit 1; }
    if $DEBUG; then
        echo "Created file '$FILENAME' with content 'File$i.txt'"
    fi
done

subha@SubhaRanjanSahoo MINGW64 ~
$ ./text.sh
Directory 'TestDir' already exists.
Created file 'TestDir/File1.txt' with content 'File1.txt'
Created file 'TestDir/File2.txt' with content 'File2.txt'
Created file 'TestDir/File3.txt' with content 'File3.txt'
Created file 'TestDir/File4.txt' with content 'File4.txt'
Created file 'TestDir/File5.txt' with content 'File5.txt'
Created file 'TestDir/File6.txt' with content 'File6.txt'
Created file 'TestDir/File7.txt' with content 'File7.txt'
Created file 'TestDir/File8.txt' with content 'File8.txt'
Created file 'TestDir/File9.txt' with content 'File9.txt'
Created file 'TestDir/File10.txt' with content 'File10.txt'
```

Assignment 6

Q. 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.

Answer:

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

```
LOGFILE="sample.log"
```

```
grep "ERROR" "$LOGFILE" | awk '{print $1, $2, $3, $4, $5, $6, $7, $8, $9}'
```

Assignment 7

Q. 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.

Answer:

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

```
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_FILE="output_${INPUT_FILE}"
```

```
sed "s/$OLD_TEXT/$NEW_TEXT/g" "$INPUT_FILE" > "$OUTPUT_FILE"
```

```
echo "Replaced all occurrences of '$OLD_TEXT' with '$NEW_TEXT' in '$INPUT_FILE' and saved the result to '$OUTPUT_FILE'."
```

Output:

Replaced all occurrences of 'old_text' with 'new_text' in 'myfile.txt' and saved the result to 'output_myfile.txt'.

-----END-----