Assignment 7

Name - Md. Firoze Baba

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.sh"

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

o/p :

```
"file.sh" [New] 8L, 123B written
[root@localhost ~]# chmod u+x myfile.sh
[root@localhost ~]# bash myfile.sh
File exists.
```

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 stop):"

while :
do
   read -p "> " num

   if [ "$num" == "0" ]; then
      break
   fi

   remainder=$((num % 2))

   if [ "$remainder" == "0" ]; then
```

```
        echo "$num is even"
    else
        echo "$num is odd"
    fi
done

echo "Program terminated."
chmod u+x odd_even.sh
sh -x odd_even.sh

Output:
Enter a number(enter 0 to stop)
6
6 is even
Enter a number(enter 0 to stop)
17
17 is odd
Enter a number(enter 0 to stop)
35
35 is odd
Enter a number(enter 0 to stop)
0
[root@localhost ~]#
```

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 "The file '$filename' has $num_lines lines."
    else
        echo "Error: File '$filename' not found."
    fi
}

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

```
count_lines "non_existent_file.txt"
chmod u+x count_lines.sh
```
Output:
```
[root@localhost ~]# bash count.sh hello.txt
The file count_lines.sh' has 12 lines
```

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

mkdir TestDir
cd TestDir

for i in {1..10}
do
   filename="File$i.txt"
   echo "$filename" > "$filename"
done

echo "Files created successfully in TestDir directory."
chmod u+x create_files.sh
```

Output:

```
[root@localhost ~]# bash create_files.sh


[root@localhost ~]# ls
odd_even.sh  bench.py  count_lines.sh  dir.sh  ex.txt  hello.c  hello.txt  TestDir


[root@localhost TestDir]# ls
File10.txt  File2.txt  File4.txt  File6.txt  File8.txt
File1.txt   File3.txt  File5.txt  File7.txt  File9.txt
```

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_mode=false

if [[ "$1" == "--debug" ]]; then
   debug_mode=true
fi

print_debug() {
   if $debug_mode; then
      echo "[DEBUG] $1"
   fi
}

dir_name="TestDir"
if [ -d "$dir_name" ]; then
   print_debug "Directory '$dir_name' already exists."
   read -p "Remove existing directory '$dir_name'? (y/n) " choice
   if [[ "$choice" =~ ^[Yy]$ ]]; then
      rm -r "$dir_name"
      print_debug "Removed existing directory '$dir_name'."
   else
      echo "Exiting script."
      exit 1
   fi
fi

mkdir "$dir_name" || { echo "Error: Failed to create directory '$dir_name'."; exit 1; }
print_debug "Created directory '$dir_name'."

cd "$dir_name" || { echo "Error: Failed to change directory to '$dir_name'."; exit 1; }
print_debug "Changed current directory to '$dir_name'."

for i in {1..10}
do
   filename="File$i.txt"
   echo "$filename" > "$filename" 2>/dev/null || { echo "Error: Failed to create file '$filename'.";
exit 1; }
   print_debug "Created file '$filename'."
```

done

echo "Files created successfully in '$dir_name' directory."
chmod u+x create_files.sh
Output:

[root@localhost ~]# bash create_files.sh
Error: Directory 'TestDir' already exits.

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
# 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

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

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"

if [ ! -f "$input_file" ]; then
    echo "Error: Input file '$input_file' not found."
    exit 1
fi

output_file="${input_file/.txt/_modified.txt}"

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

echo "Text replacement completed. Output saved to $output_file."
chmod u+x replace_text.sh
sh -x replace_text.sh
```

Output:

```
[root@localhost ~]# bash replace_text firoze
Replace done. result stored to input_modified.txt
```