Assignment 05:
Modify the script to handle errors, such as the directory already existing or lacking permissions to create files. Add a debugging mode that printsadditional information when enabled.

Handling Errors:

- Check if the directory already exists before attempting to create it.
- If it exists, you can eitherabort the script or prompt the user for further action.
- Check for permissions to create files in the specified directory.
- If the script lacks the necessarypermissions, it should gracefully handle the error, informing the user and possibly exiting withan error code.

Debugging Mode:

- Add an option to enable debugging mode, which will print additional information during scriptexecution.
- This can include variable values, intermediate steps, or any other information that helps in understanding the script's behavior.

```
#!/bin/bash

# Function to print debug

messagesdebug() {

    if [ "$DEBUG" = true ];

        thenecho "DEBUG:

        $1"

    fi

}


# Function to handle

errorshandle_error() {

    echo "Error:

    $1"exit 1

}
```

```
# Parse command line
argumentswhile getopts ":d"
opt; do
        case ${opt}
                ind )
                        DEBUG=true
                        ;;
                \? )
                        echo "Usage: $0 [-d (enable
                        debugging)]"exit 1
                        ;;
        esac
 done
 shift $((OPTIND -1))
 # Check if directory already
 existsif [ -d "$1" ]; then
        handle_error "Directory already exists. Aborting."
 fi
 # Check permissions to create
 filesif [ ! -w "$1" ]; then
        handle_error "Insufficient permissions to create files in the specified directory."
 Fi
 # Create the directory
 mkdir -p "$1" || handle_error "Failed to create
 directory."debug "Directory created successfully."
 # Add more script logic here...
 echo "Script execution completed successfully."
```

In this script:

- The debug function prints messages if debugging mode is enabled.
- The handle_error function prints an error message and exits the script with a non-zero exitcode.
- Command-line options are parsed using getopts. -d enables debugging mode.The existence of the directory is checked using [-d "$1"].

To use the script, you would run it like this:

./your_script.sh -d /path/to/directory

This will enable debugging mode and attempt to create the specified directory.

Assignment 06:

Given a sample log file, write a script using grep to extract all linescontaining "ERROR". Use awk to print the date, time, and error message of each extracted line.

Data Processing with sed

#!/bin/bash

Sample log file

logfile="sample.log"

Use grep to extract lines containing

"ERROR"error_lines=$(grep "ERROR"

"$logfile")

Use awk to print date, time, and error message of each extracted line

echo "$error_lines" | awk '{print $1, $2, $NF}'

In this script:

We specify the path to the sample log file.

grep "ERROR" "$logfile" extracts lines containing "ERROR" from the log file.

awk '{print $1, $2, $NF}' prints the first and second fields (date and time) along with the lastfield (error message) of each extracted line.

sed 's/^[0-9]\{4\}-[0-9]\{2\}-[0-9]\{2\} [0-9]\{2\}:[0-9]\{2\}:[0-9]\{2\}//' removes the timestampfrom each error message. Adjust the regular expression as needed for your timestamp format.

To use the script, save it to a file (e.g., extract_errors.sh), make it executable (chmod +x extract_errors.sh), and run it:

```
./extract_errors.sh
```

This script will extract lines containing "ERROR" from the sample log file, print the date, time, and error message of each extracted line using awk, and perform additional data processing onerror messages using sed.

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

# Check if the correct number of arguments is providedif

[ "$#" -ne 3 ]; then

    echo "Usage: $0 input_file old_text new_text"

    exit 1

fi

# Assign input file and old/new text

input_file="$1"

old_text="$2"

new_text="$3"

# Check if input file exists

if [ ! -f "$input_file" ]; then
```

```
    echo "Error: Input file '$input_file' not found."
    exit 1
fi


 # Perform the replacement using sed and output to a new file
 output_file="${input_file}.new"
 sed "s/$old_text/$new_text/g" "$input_file" > "$output_file"
 echo "Replacement complete. Result written to '$output_file'."
```

To use this script:

Save the script to a file (e.g., replace_text.sh).

Make it executable: chmod +x replace_text.sh.

Run it with three arguments: the input file, old text, and new text:

```
./replace_text.sh input_file.txt old_text new_text
```

This script checks if the correct number of arguments is provided, ensures the input file exists, performs the replacement using sed, and outputs the result to a new file