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".

#!/bin/bash

file="myfile.txt"

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.

#!/bin/bash

echo "Enter numbers (enter 0 to exit):"

while true; do

read -p "> " num

if [ $num -eq 0 ]; then

echo "Exiting..."

break

elif [ $((num % 2)) -eq 0 ]; then

echo "$num is even."

else

echo "$num is odd."

fi

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.

#!/bin/bash

# Define a function to count lines in a file

count\_lines() {

filename=$1

if [ -f "$filename" ]; then

num\_lines=$(wc -l < "$filename")

echo "Number of lines in $filename: $num\_lines"

else

echo "File $filename not found."

fi

}

# Call the function with different filenames

count\_lines "file1.txt"

count\_lines "file2.txt"

count\_lines "file3.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").

#!/bin/bash

# Create TestDir if it doesn't exist

mkdir -p TestDir

# Change to TestDir directory

cd TestDir || exit

# Loop to create ten files

for ((i=1; i<=10; i++)); do

filename="File${i}.txt"

echo "$filename" > "$filename"

done

echo "Files created in TestDir:"

ls -l

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.

#!/bin/bash

# Function to print debug messages

debug() {

if [ "$debug\_mode" = true ]; then

echo "DEBUG: $1"

fi

}

# Set debugging mode based on argument

if [ "$1" = "-d" ]; then

debug\_mode=true

else

debug\_mode=false

fi

# Create TestDir if it doesn't exist

if [ ! -d "TestDir" ]; then

debug "Creating directory TestDir..."

mkdir -p TestDir || { echo "Error: Unable to create directory TestDir."; exit 1; }

else

debug "TestDir already exists."

fi

# Change to TestDir directory

debug "Changing to TestDir directory..."

cd TestDir || { echo "Error: Unable to change to TestDir directory."; exit 1; }

# Loop to create ten files

for ((i=1; i<=10; i++)); do

filename="File${i}.txt"

debug "Creating file $filename..."

echo "$filename" > "$filename" || { echo "Error: Unable to create file $filename."; exit 1; }

done

echo "Files created in TestDir:"

ls -l

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

#!/bin/bash

# Using grep to extract lines containing "ERROR"

grep "ERROR" sample.log | \

# Using awk to print date, time, and error message

awk '{

# Extracting date and time fields

match($0, /^[A-Za-z]+ [0-9]+ [0-9]{2}:[0-9]{2}:[0-9]{2}/)

datetime = substr($0, RSTART, RLENGTH)

# Extracting error message

match($0, /ERROR.\*/)

error\_message = substr($0, RSTART, RLENGTH)

# Print date, time, and error message

print datetime, error\_message

}'

#!/bin/bash

# Using grep to extract lines containing "ERROR"

grep "ERROR" sample.log | \

# Using awk to print date, time, and error message

awk '{

# Extracting date and time fields

match($0, /^[A-Za-z]+ [0-9]+ [0-9]{2}:[0-9]{2}:[0-9]{2}/)

datetime = substr($0, RSTART, RLENGTH)

# Extracting error message

match($0, /ERROR.\*/)

error\_message = substr($0, RSTART, RLENGTH)

# Print date, time, and error message

print datetime, error\_message

}' | \

# Using sed to replace "ERROR" with "WARNING" in error messages

sed 's/ERROR/WARNING/g'

Assignment7: 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 are provided

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="${input\_file%.txt}\_replaced.txt"

# Check if the input file exists

if [ ! -f "$input\_file" ]; then

echo "Error: Input file '$input\_file' not found."

exit 1

fi

# Use sed to replace old\_text with new\_text and write to output\_file

sed "s/$old\_text/$new\_text/g" "$input\_file" > "$output\_file"

echo "Replacement complete. Result saved to $output\_file."

./replace\_text.sh input\_file.txt old\_text new\_text

./replace\_text.sh fruits.txt apple orange