Shell Scripting with Bash

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
vi file.sh
file_name=”myfile.txt”
if [ -e “$file_name” ]; 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
# Function to check if a number is odd or even
check_odd_or_even() {
   if [ $(($1 % 2)) -eq 0 ]; then
      echo "$1 is even"
   else
      echo "$1 is odd"
   fi
}
# Loop to read numbers from the user
while true; do
   read -p "Enter a number (0 to quit): " number
   if [ "$number" -eq 0 ]; then
      echo "Exiting..."
      break
   fi
```

```
    check_odd_or_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.

```
#!/bin/bash

# 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").

```
#!/bin/bash

# Create a directory named Test
mkdir Test

# Change to the Test directory
cd Test || exit

# Create ten files with the specified content
for ((i = 1; i <= 10; i++)); do
    filename="File$i.txt"
    echo "$filename" > "$filename"
    echo "Created $filename with content: $filename"
done
```

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

# Debugging mode flag
debug_mode=false

# Function for debug messages
debug() {
    if [ "$debug_mode" = true ]; then
```

```
        echo "Debug: $1"
    fi
}

# Function to create directory and files
create_files() {
    local directory_name="$1"
    debug "Creating directory: $directory_name"
    if [ -d "$directory_name" ]; then
        echo "Error: Directory '$directory_name' already exists."
        exit 1
    fi

    mkdir -p "$directory_name"
    if [ $? -ne 0 ]; then
        echo "Error: Failed to create directory '$directory_name'."
        exit 1
    fi

    debug "Changing to directory: $directory_name"
    cd "$directory_name" || { echo "Error: Unable to change to directory '$directory_name'"; exit 1; }

    for ((i = 1; i <= 10; i++)); do
        filename="File$i.txt"
        debug "Creating file: $filename"
        echo "$filename" > "$filename" || { echo "Error: Failed to create file '$filename'"; exit 1; }
        echo "Created $filename with content: $filename"
    done
}
```

```
# Check for debugging mode
if [ "$1" = "--debug" ]; then
    debug_mode=true
    debug "Debug mode enabled."
fi

# Call function to create files
create_files "TestDir"
```

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

# Extract lines containing "ERROR" using grep
grep "ERROR" sample.log > error_lines.txt

# Use awk to print date, time, and error message from each line
awk '{print $1, $2, $NF}' error_lines.txt | sed 's/^\([^ ]*\) \([^ ]*\) \(.*\)$/\1 \2 \3/'
```

explanation

grep "ERROR" sample.log > error_lines.txt: This command searches for lines containing "ERROR" in the sample.log file and redirects the output to a new file named error_lines.txt.

awk '{print $1, $2, $NF}' error_lines.txt: This awk command prints the first field (date), second field (time), and last field (error message) from each line in error_lines.txt. $1, $2, and $NF are used to refer to the specific fields.

sed 's/^\([^]*\) \([^]*\) \(.*\)$/\1 \2 \3/': This sed command is used to format the output. It captures the date, time, and error message using regular expressions and rearranges them with spaces between each component.

When you run this script, it will extract all lines containing "ERROR" from sample.log, then use awk to extract the date, time, and error message from each line, and finally format the output 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 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"

# Check if the 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 write the result to a new file
output_file="modified_$input_file"
sed "s/$old_text/$new_text/g" "$input_file" > "$output_file"

echo "Replacement done. Output saved to '$output_file'."
```

```
 chmod +x replace_text.sh.
./replace_text.sh input_file.txt old_text new_text
```