Green University of Bangladesh

Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: (Summer,Year:2025),B.Sc.in CSE (Day)

LAB REPORT NO - 02

Course Title: Operating System Lab

Course Code: CSE 310 Section: 223-D2

Lab Experiment Name :Lab Manuals -04 Lab Exercise

Student Details

	Name	ID
1.	MD.SHAJALAL	223002088

Submission Date : 09 – 08 - 2025

Course Teacher's Name : Umme Habiba

Lab Report Status	
Marks: ……	Signature:.....
Comments:.....	Date:.....

Objective :

The objective of this lab is to gain practical knowledge of **shell scripting** by implementing various programs that use loops, arrays, functions, and string processing.
Specifically, the experiments aim to:

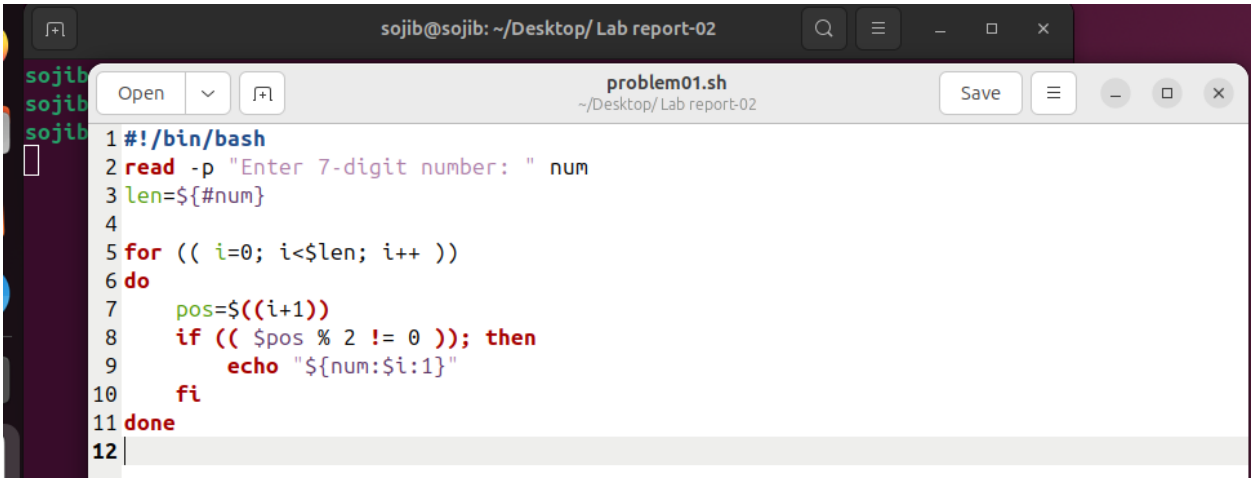
1. Extract digits from specific positions in a number using a for loop.
2. Count occurrences of each digit using a while loop and associative arrays.
3. Identify the 2nd and 3rd largest numbers from a list and compute their sum using array manipulation.
4. Calculate the factorial of two numbers using user-defined functions and determine their sum.
5. Count the total number of alphabets, digits, and special characters in a string using pattern matching.

Q.1. Write a shell program to display odd position numbers (using For loop).

Algorithm:

1. Read a 7-digit number from the user.
2. Calculate the length of the number.
3. Use a for loop to iterate over each digit.
4. Check if the position is odd.
5. Print digits at odd positions.

Code:



Sample Output:

```
sojib@sojib: ~/Desktop/ Lab report-02
sojib@sojib:~/Desktop/ Lab report-02$ touch problem01.sh
sojib@sojib:~/Desktop/ Lab report-02$ chmod +x problem01.sh
sojib@sojib:~/Desktop/ Lab report-02$ gedit problem01.sh
^Z
[1]+  Stopped                 gedit problem01.sh
sojib@sojib:~/Desktop/ Lab report-02$ ./problem01.sh
Enter 7-digit number: 5867458
5
6
4
8
sojib@sojib:~/Desktop/ Lab report-02$
```

Q.2. Write a Shell program using while loop:

Algorithm:

1. Read a number as a string.
2. Initialize an associative array to store counts.
3. Use a while loop to process each character.
4. Increment the count for each digit found.
5. Display the counts.

Code:

```
Open   problem02.sh   Save
~/Desktop/ Lab report-02
#!/bin/bash
read -p "Enter the number: " num
declare -A count

while [ -n "$num" ]
do
    digit=${num:0:1}
    count[$digit]=$(( ${count[$digit]} + 1 ))
    num=${num:1}
done

for key in "${!count[@]}"
do
    echo "$key = ${count[$key]} times"
done
```

Sample Output:

```
sojib@sojib:~/Desktop/ Lab report-02$ gedit problem02.sh
^Z
[1]+  Stopped                 gedit problem02.sh
sojib@sojib:~/Desktop/ Lab report-02$ ./problem02.sh
Enter the number: 148541547854
8 = 2 times
7 = 1 times
5 = 3 times
4 = 4 times
1 = 2 times
sojib@sojib:~/Desktop/ Lab report-02$
```

Q.3. Write a Shell program to find the 2nd highest and 3rd highest numbers from a set of numbers and sum of them using array.

Algorithm:

1. Read n numbers from the user.
2. Store them in an array.
3. Sort the array in descending order.
4. Extract the 2nd and 3rd largest numbers.
5. Calculate and display their sum.

Code:

Open | *problem03.sh ~/Desktop/ Lab report-02 | Save

```bash
#!/bin/bash
read -p "Enter the number of elements: " n
for (( i=0; i<n; i++ ))
do
    read -p "Enter the number: " arr[$i]
done

sorted=($(printf "%s\n" "${arr[@]}" | sort -nr))

second=${sorted[1]}
third=${sorted[2]}
sum=$((second + third))

echo "The sum of second and third highest numbers is: ($second + $third) = $sum"
```

Sample Output:

```
sojib@sojib:~/Desktop/ Lab report-02$ ./problem03.sh
Enter the number of elements: 5
Enter the number: 10
Enter the number: 21
Enter the number: 30
Enter the number: 17
Enter the number: 5
The sum of second and third highest numbers is: (21 + 17) = 38
```

Q.4. Write a Shell program to find the factorial of two different numbers and sum of the numbers using function.

Algorithm:

1. Define a function factorial to calculate factorial.
2. Call the function for each number.
3. Add the results.
4. Display the factorials and the sum.

Code:

Open problem04.sh ~/Desktop/ Lab report-02 Save

```
#!/bin/bash
factorial() {
    num=$1
    fact=1
    for (( i=1; i<=num; i++ ))
    do
        fact=$((fact * i))
    done
    echo $fact
}

f1=$(factorial 5)
f2=$(factorial 6)

echo "Factorial of 5 is $f1"
echo "Factorial of 6 is $f2"
sum=$((f1 + f2))
echo "$f1 + $f2 = $sum"
```

Sample Output:

```
sojib@sojib:~/Desktop/ Lab report-02$ ./problem04.sh
Factorial of 5 is 120
Factorial of 6 is 720
120 + 720 = 840
sojib@sojib:~/Desktop/ Lab report-02$
```

Q.5. Write a Shell program to find total number of alphabets, digits or special characters in a string.

Algorithm:

1. Read the string from the user.
2. Initialize counters for alphabets, digits, and special characters.
3. Loop through each character and classify it.
4. Display the counts.

Code:

problem05.sh
~/Desktop/ Lab report-02

```bash
#!/bin/bash
read -p "Enter a string: " str
alpha=0
digit=0
special=0

for (( i=0; i<${#str}; i++ ))
do
    ch=${str:$i:1}
    if [[ $ch =~ [A-Za-z] ]]; then
        alpha=$((alpha+1))
    elif [[ $ch =~ [0-9] ]]; then
        digit=$((digit+1))
    else
        special=$((special+1))
    fi
done

echo "Alphabets = $alpha"
echo "Digits = $digit"
echo "Special characters = $special"
```

Sample Output:

```
sojib@sojib:~/Desktop/ Lab report-02$ ./problem05.sh
Enter a string: Today is 12 November
Alphabets = 15
Digits = 2
Special characters = 3
sojib@sojib:~/Desktop/ Lab report-02$
```

Conclusion :

From these experiments, it can be concluded that shell scripting is a powerful and flexible tool for automating tasks and processing data in UNIX/Linux environments.
The key takeaways include:

- Looping constructs (for and while) are essential for iterating over strings, numbers, and arrays.
- String indexing in bash allows easy extraction of specific characters or substrings.
- Associative arrays can be used to efficiently store and retrieve frequency counts.
- Sorting techniques combined with arrays enable quick selection of maximum or specific-order elements.
- Functions promote code reusability and make programs easier to maintain.
- Pattern matching with regular expressions helps classify characters in strings accurately.