Experiment [6] : [Shell Programming- Loops]

Name: Prapti Uniyal Roll.: 590028360 Date: 15-09-2025

AIM:

- [To Learn usage of loops in Bash Scripting]

Requirements:

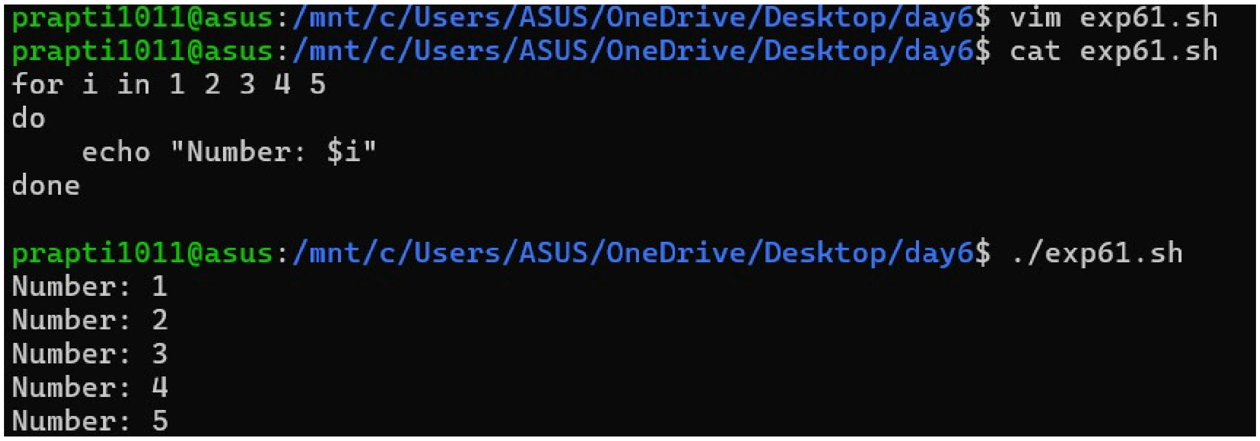
- [Any Linux Distro, any kind of text editor (vs code, vim, nano, etc)]

Theory:

- Basic usage of loops to execute a block of code repeatedly.

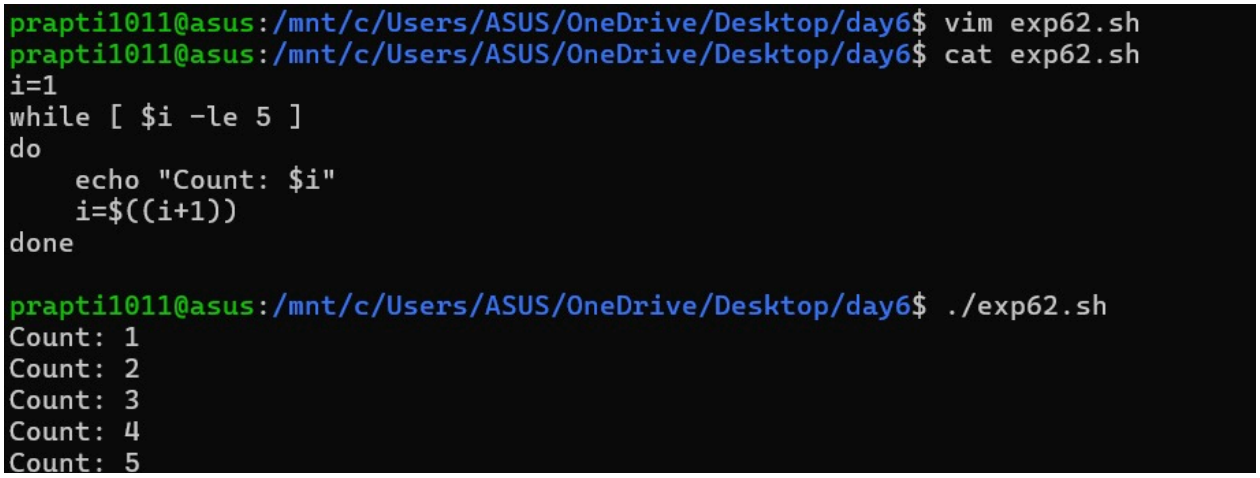
Loops:

1). for loop syntax and execution-



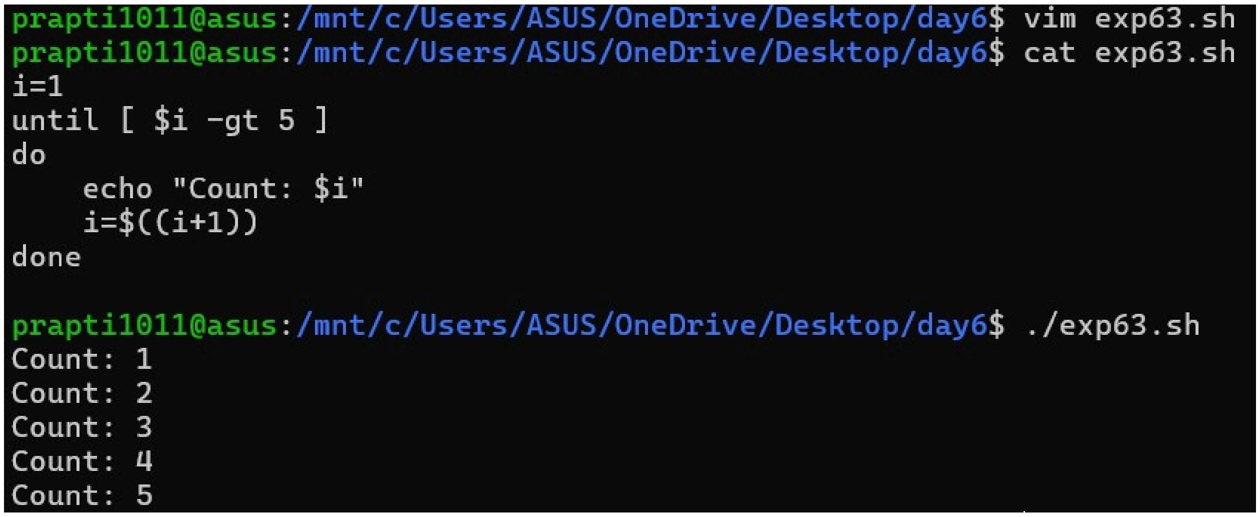
```
prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6$ vim exp61.sh
prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6$ cat exp61.sh
for i in 1 2 3 4 5
do
    echo "Number: $i"
done

prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6$ ./exp61.sh
Number: 1
Number: 2
Number: 3
Number: 4
Number: 5
```

2). while loop syntax and execution-



```
prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6$ vim exp62.sh
prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6$ cat exp62.sh
i=1
while [ $i -le 5 ]
do
    echo "Count: $i"
    i=$((i+1))
done

prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6$ ./exp62.sh
Count: 1
Count: 2
Count: 3
Count: 4
Count: 5
```

3). until loop syntax and execution-

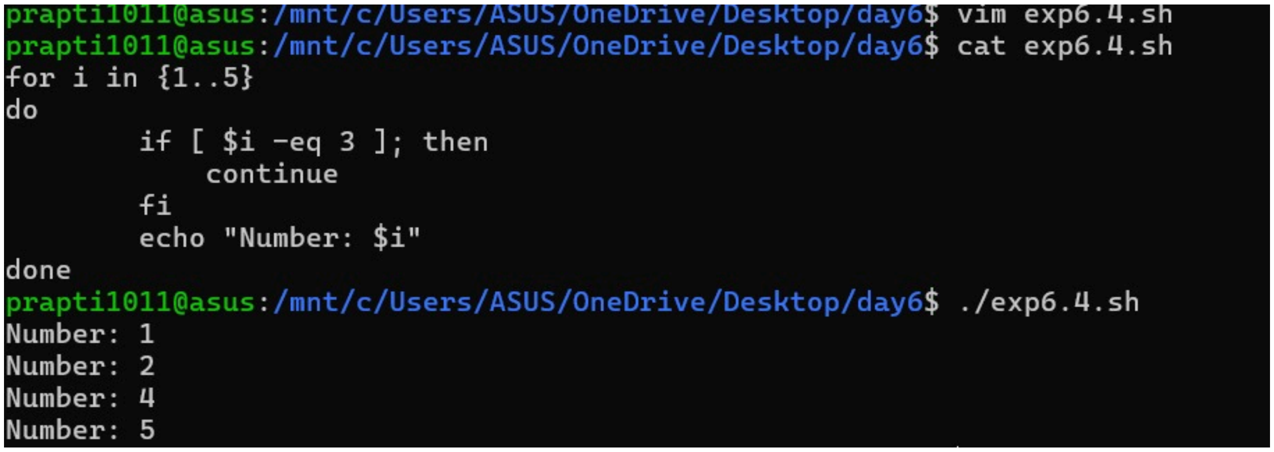


```
prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6$ vim exp63.sh
prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6$ cat exp63.sh
i=1
until [ $i -gt 5 ]
do
    echo "Count: $i"
    i=$((i+1))
done

prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6$ ./exp63.sh
Count: 1
Count: 2
Count: 3
Count: 4
Count: 5
```

Loop Controls:

1). break- it is a control flow keyword used to immediately terminate the currently executing loop.

2). continue- this keyword is a control statement used inside loops to skip the rest of the current iteration and immediately start the next one.

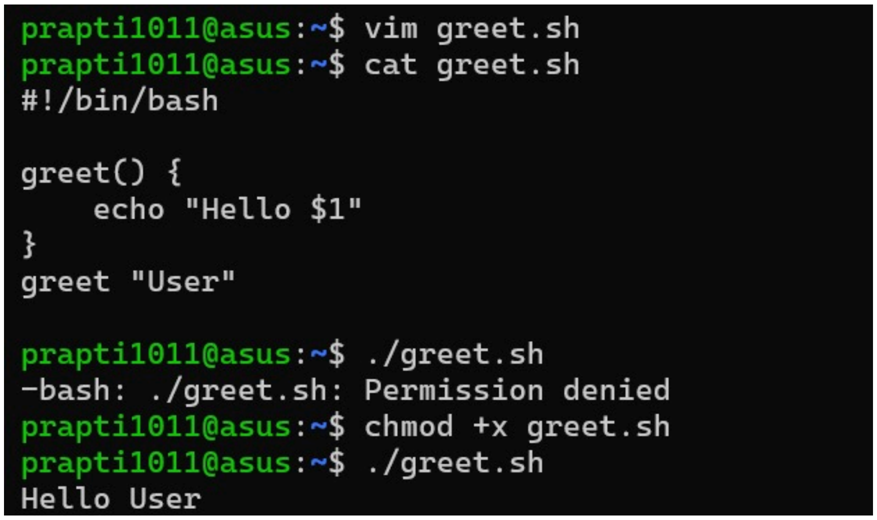


```
prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6$ vim exp6.4.sh
prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6$ cat exp6.4.sh
for i in {1..5}
do
        if [ $i -eq 3 ]; then
            continue
        fi
        echo "Number: $i"
done
prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6$ ./exp6.4.sh
Number: 1
Number: 2
Number: 4
Number: 5
```

Shell functions:

Shell functions are named blocks of commands within a shell script designed for reuseability and organization.

Example-



```
prapti1011@asus:~$ vim greet.sh
prapti1011@asus:~$ cat greet.sh
#!/bin/bash

greet() {
    echo "Hello $1"
}
greet "User"

prapti1011@asus:~$ ./greet.sh
-bash: ./greet.sh: Permission denied
prapti1011@asus:~$ chmod +x greet.sh
prapti1011@asus:~$ ./greet.sh
Hello User
```

Input/Output redirection:

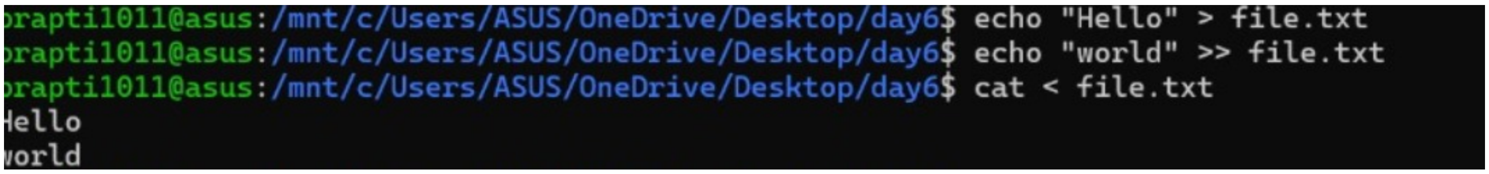
It is a shell feature that allows alteration of the default input and output streams of commands.

1). Output redirection:

a) > : overwrite b) >> : append c) 2> : redirect errors

2). Input redirection:

a) < : input from file

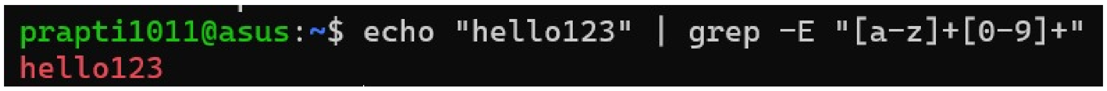


```
prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6$ echo "Hello" > file.txt
prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6$ echo "world" >> file.txt
prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6$ cat < file.txt
Hello
world
```

Regular expressions:

These are powerful text patterns used for searching,matching,and manipulating text.

A comman linux command for searching test based on a regular expression pattern within files- grep

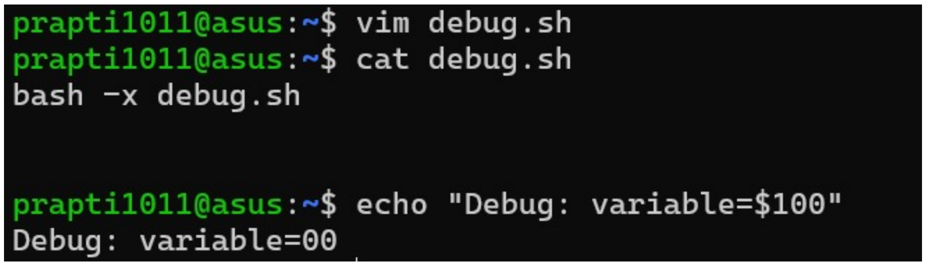


```
prapti1011@asus:~$ echo "hello123" | grep -E "[a-z]+[0-9]+"
hello123
```

Script debugging and troubleshooting:

It involves identifying and resolving issues within scripts to ensure their correcr functionality.

While troubleshooting generally focuses on identifying system- level issues , debugging specifically addressws errors within the code itself.



```
prapti1011@asus:~$ vim debug.sh
prapti1011@asus:~$ cat debug.sh
bash -x debug.sh


prapti1011@asus:~$ echo "Debug: variable=$100"
Debug: variable=00
```

Procedure & Observations:

Task [1]:[Palindrome Check]

Task Statement:

- [To check if the number given by the user is a palindrome number or not.]

Explanation:

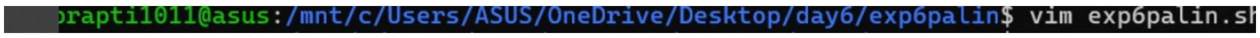
- [using while loop wap to check if the number is a palindrome or not.]

Command(s):

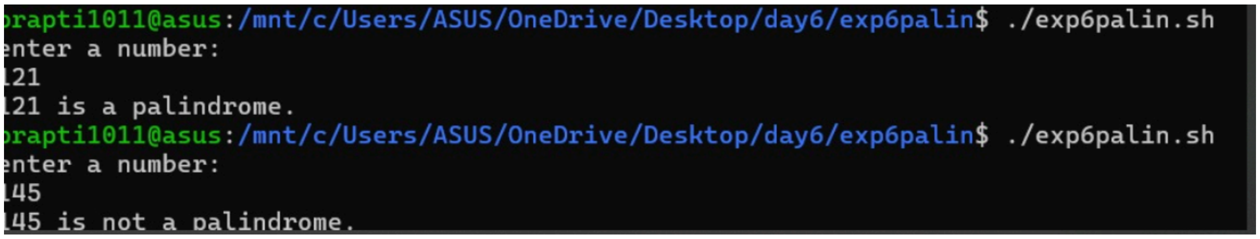
```
#!/bin/bash
echo "Enter a number: "
read num
```

```
rev=0
temp=$num

while [ $temp -gt 0 ]
do
    digit=$((temp % 10))
    rev=$((rev * 10 + digit))
    temp=$((temp / 10))
done

if [ $num -eq $rev ]
then
    echo "$num is a palindrome."
else
    echo "$num is not a palindrome."
fi
```

Output:



```
prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6/exp6palin$ vim exp6palin.sh
```



```
prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6/exp6palin$ ./exp6palin.sh
nter a number:
121
121 is a palindrome.
prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6/exp6palin$ ./exp6palin.sh
nter a number:
145
145 is not a palindrome.
```

Task [2]: [Finding GCD & LCM]

Task Statement:

- [Take two inputs from user and find the gcd and lcm of the two inputs.]

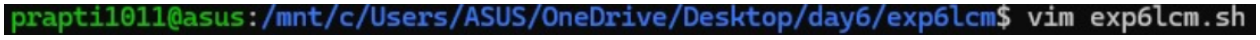
Explanation:

- [This script will take input from user and will give the following output.]

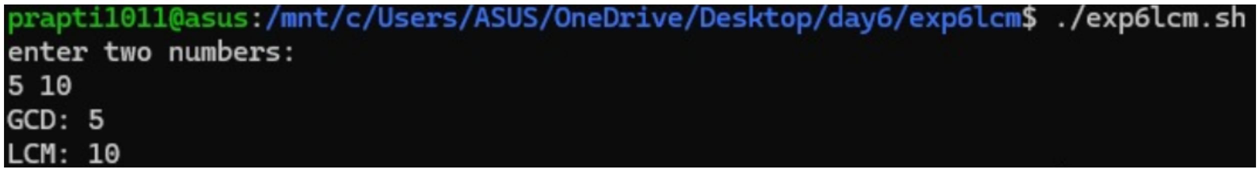
Command(s):

```
#!/bin/bash
echo "Enter two numbers: "
read a b

x=$a
y=$b
while [ $y -ne 0 ]
do
    temp=$y
    y=$((x % y))
    x=$temp
done
gcd=$x

lcm=$(( (a * b) / gcd ))

echo "GCD: $gcd"
echo "LCM: $lcm"
```

Output:



```
praptil011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6/exp6lcm$ vim exp6lcm.sh
```



```
praptil011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6/exp6lcm$ ./exp6lcm.sh
enter two numbers:
5 10
GCD: 5
LCM: 10
```

Task [3]: [Sorting numbers]

Task Statement:

- [Input numbers separated by space and sort them in ascending and descending order]

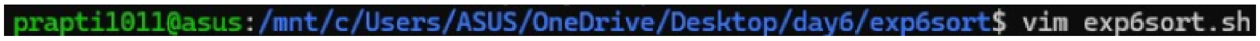
Explanation:

- [This script will compare the input values and sort them accordingly]

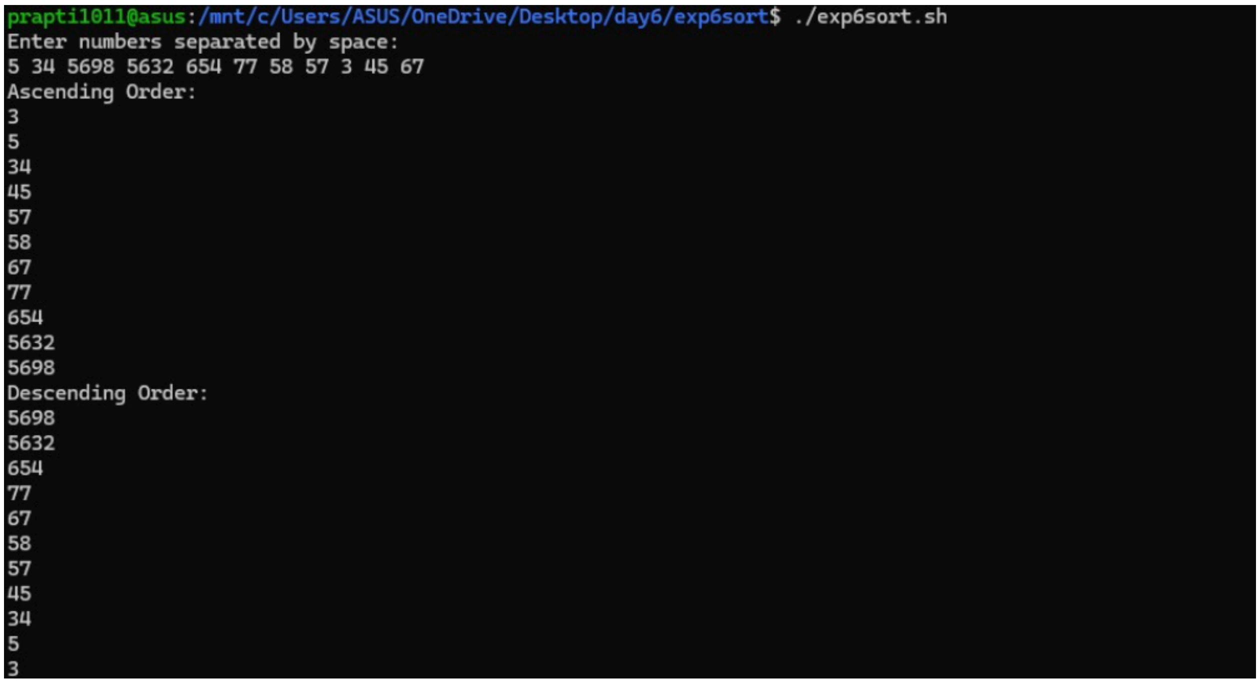
Command(s):

```
#!/bin/bash
echo "Enter numbers separated by space: "
read -a arr

echo "Ascending Order: "
printf "%s\n" "${arr[@]}" | sort -n

echo "Descending Order: "
printf "%s\n" "${arr[@]}" | sort -nr
```

Output:



```
praptil011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6/exp6sort$ vim exp6sort.sh
```



```
praptil011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6/exp6sort$ ./exp6sort.sh
Enter numbers separated by space:
5 34 5698 5632 654 77 58 57 3 45 67
Ascending Order:
3
5
34
45
57
58
67
77
654
5632
5698
Descending Order:
5698
5632
654
77
67
58
57
45
34
5
3
```

ASSIGNMENTS

Exercise [1] : [Factorial of a number]

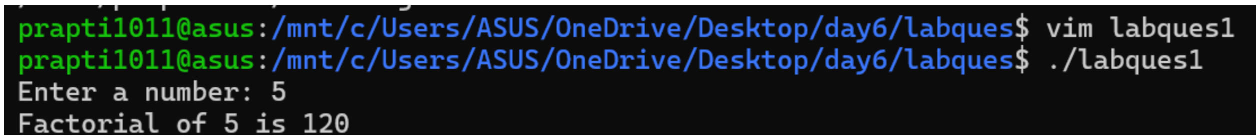
Task Statement:

- [Write a function to calculate the factorial of a number using a loop]

Command(s):

```
#!/bin/bash
read -p "Enter a number: " num

if ! [[ "$num" =~ ^[0-9]+$ ]]; then
    echo "Invalid input. Please enter a non-negative integer."
    exit 1
fi

factorial=1
for (( i=1; i<=num; i++ ))
do
    factorial=$((factorial * i))
done
echo "Factorial of $num is $factorial"
```

Output:



```
prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6/labques$ vim labques1
prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6/labques$ ./labques1
Enter a number: 5
Factorial of 5 is 120
```

Exercise [2]: [Counts appearances of a word in file]

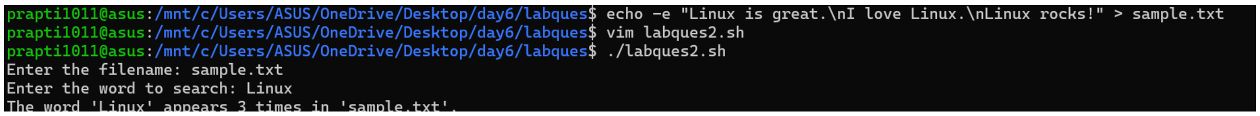
Task Statement:

- [write a script that reads a filename and counts how many times a given word appears in it]

Command(s):

```
#!/bin/bash

read -p "Enter the filename: " filename
read -p "Enter the word to search: " word

if [ ! -f "$filename" ]; then
    echo "File '$filename' does not exist."
    exit 1
fi

count=$(grep -o -w "$word" "$filename" | wc -l)

echo "The word '$word' appears $count times in '$filename'."
```

Output:



```
prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6/labques$ echo -e "Linux is great.\nI love Linux.\nLinux rocks!" > sample.txt
prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6/labques$ vim labques2.sh
prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6/labques$ ./labques2.sh
Enter the filename: sample.txt
Enter the word to search: Linux
The word 'Linux' appears 3 times in 'sample.txt'.
```

Exercise [3]: [Fibonacci series]

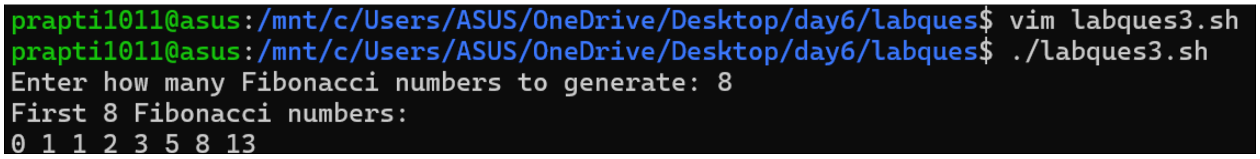
Task Statement:

- [Write a script that generates the first N Fibonacci numbers using a while loop]

Command(s):

```
#!/bin/bash

read -p "Enter how many Fibonacci numbers to generate: " n

if ! [[ "$n" =~ ^[0-9]+$ ]]; then
    echo "Please enter a valid non-negative integer."
    exit 1
fi

a=0
b=1
count=0

echo "First $n Fibonacci numbers:"

while [ $count -lt $n ]
do
    echo -n "$a "
    fn=$((a + b))
    a=$b
    b=$fn
    count=$((count + 1))
done

echo ""
```

Output:



```
prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6/labques$ vim labques3.sh
prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6/labques$ ./labques3.sh
Enter how many Fibonacci numbers to generate: 8
First 8 Fibonacci numbers:
0 1 1 2 3 5 8 13
```

<

Exercise [4]: [Check whether the entered string is proper or not]

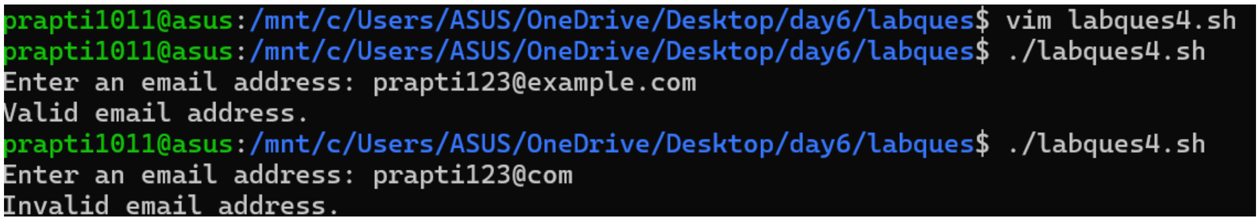
Task Statement:

- [Write a script that validates whether the entered string is a proper email address using a regular expression]

Command(s):

```
#!/bin/bash

read -p "Enter an email address: " email
pattern="^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$"
if [[ $email =~ $pattern ]]; then
    echo "Valid email address."
else
    echo "Invalid email address."
fi
```

Output:



```
prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6/labques$ vim labques4.sh
prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6/labques$ ./labques4.sh
Enter an email address: prapti123@example.com
Valid email address.
prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6/labques$ ./labques4.sh
Enter an email address: prapti123@com
Invalid email address.
```

Exercsie [5]: [Explain debug output]

Task Statement:

- [Write a script with an intentional error,run it with bash-x , and explain the debug output]

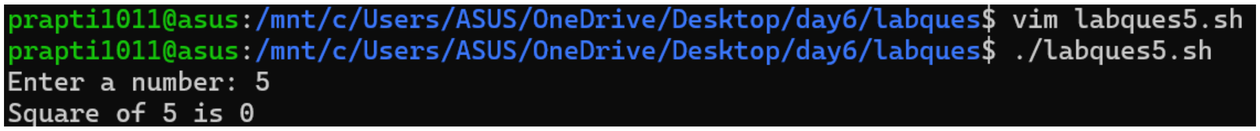
Command(s):

```
#!/bin/bash

read -p "Enter a number: " number
square=$((numbr * numbr))  # 'numbr' is undefined
echo "Square of $number is $square"
```

Output:

EXPLANATION OF DEBUG OUTPUT:

- read -p "Enter a number: " number -Bash shows the command being executed: reading input into.
- square=$((numbr * numbr)) -Bash tries to evaluate , but it's undefined — this is the intentional error.
- labques5.sh: line 5 numbr: unbound variable -Bash throws an error because was never set.
- echo 'Square of 5 is' -The final output is incorrect because couldn't be calculated.



```
prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6/labques$ vim labques5.sh
prapti1011@asus:/mnt/c/Users/ASUS/OneDrive/Desktop/day6/labques$ ./labques5.sh
Enter a number: 5
Square of 5 is 0
```

Challenges faced:

- Remembering arithmetic syntax in Bash — used `(( ))` and `expr` where needed.

Learning:

- Correct quoting and use of control constructs prevent common bugs.

Result:

- The exercises and assignments were successfully completed for shell programming using loops.