Experiment No. 5

Shell Programming – Command Line Arguments, Arrays, and Conditional Statements

Aim

To understand the use of command line arguments, arrays, and conditional statements in shell programming.

Requirements

Linux/Unix operating system

Terminal access

Bash shell

Knowledge of basic shell scripting concepts

Concept

1.Command Line Arguments

These are values provided to a shell script when it is executed.

They make a script flexible since the same script can run with different inputs.

Special variables used:

$0 → Script name

$1, $2, … → Positional arguments

$# → Total number of arguments

$@ → All arguments together

2. Arrays in Bash

An array is a variable that can hold multiple values under one name.

Elements are stored in an indexed manner.

Arrays are useful for storing lists of inputs (for example, all command line arguments).

They make scripts efficient when handling multiple values at once.

3. Conditional Statements

Conditional statements are used to make decisions in a script.

They check whether a given condition is true or false, and execute commands accordingly.

Commonly used statements:

if → Executes a block if the condition is true.

if–else → Executes one block if the condition is true, another if false.

elif → Allows multiple conditions to be checked.

These statements make scripts interactive and logical

Procedure

1. Create a shell script that accepts command line arguments

2. Pass different sets of values while running the script.

3. Store the arguments into an array for easy access and display.

4. Use conditional statements to check conditions such as:

Number of arguments given.

Whether a value matches a certain condition.

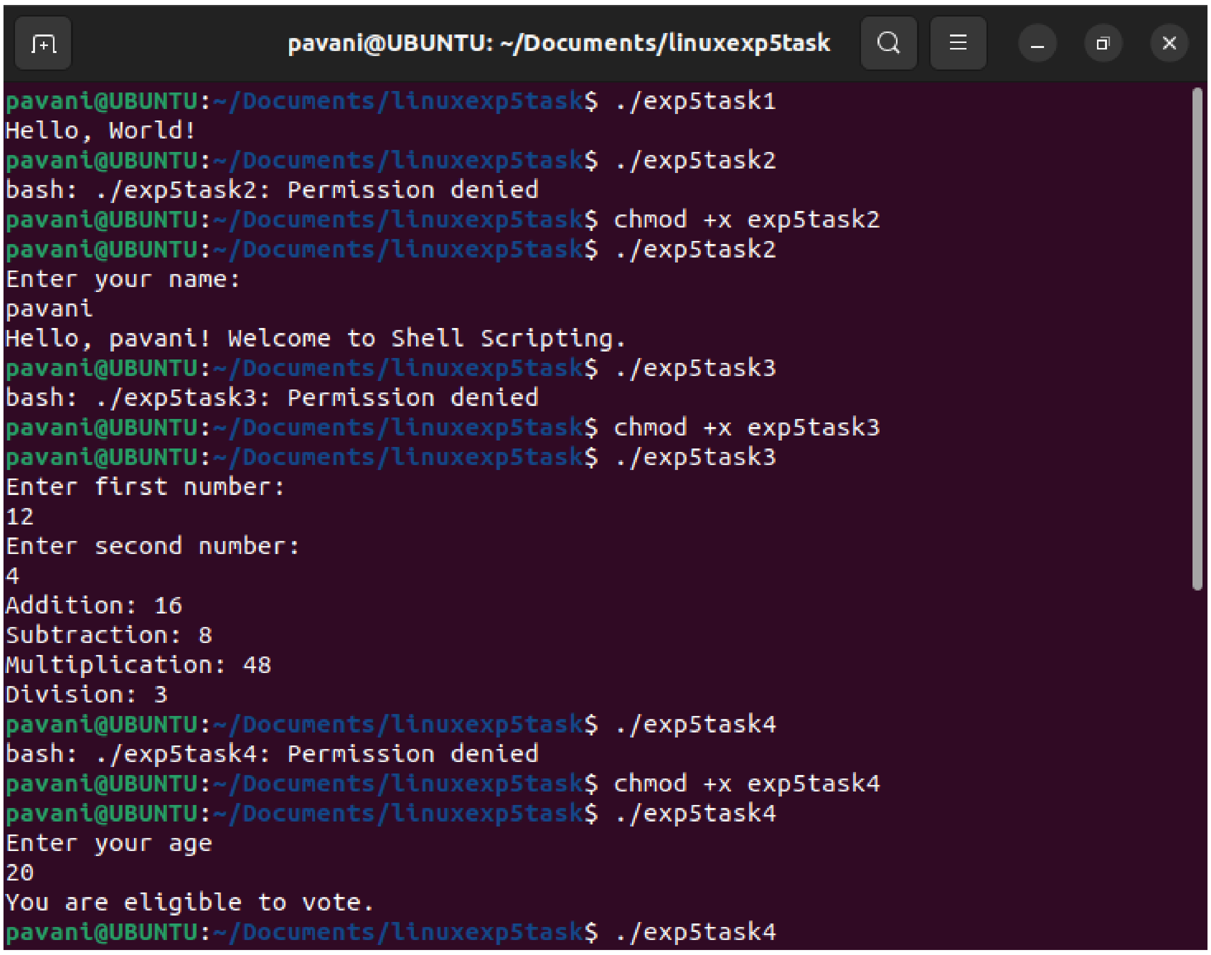
Different cases for input values.

5. Run the script with various arguments and observe the results.

LAB TASKS

TASK I,2,3,4-EXPERIMENT 4 Basics

OUTPUT OF TASK 1,2,3,4

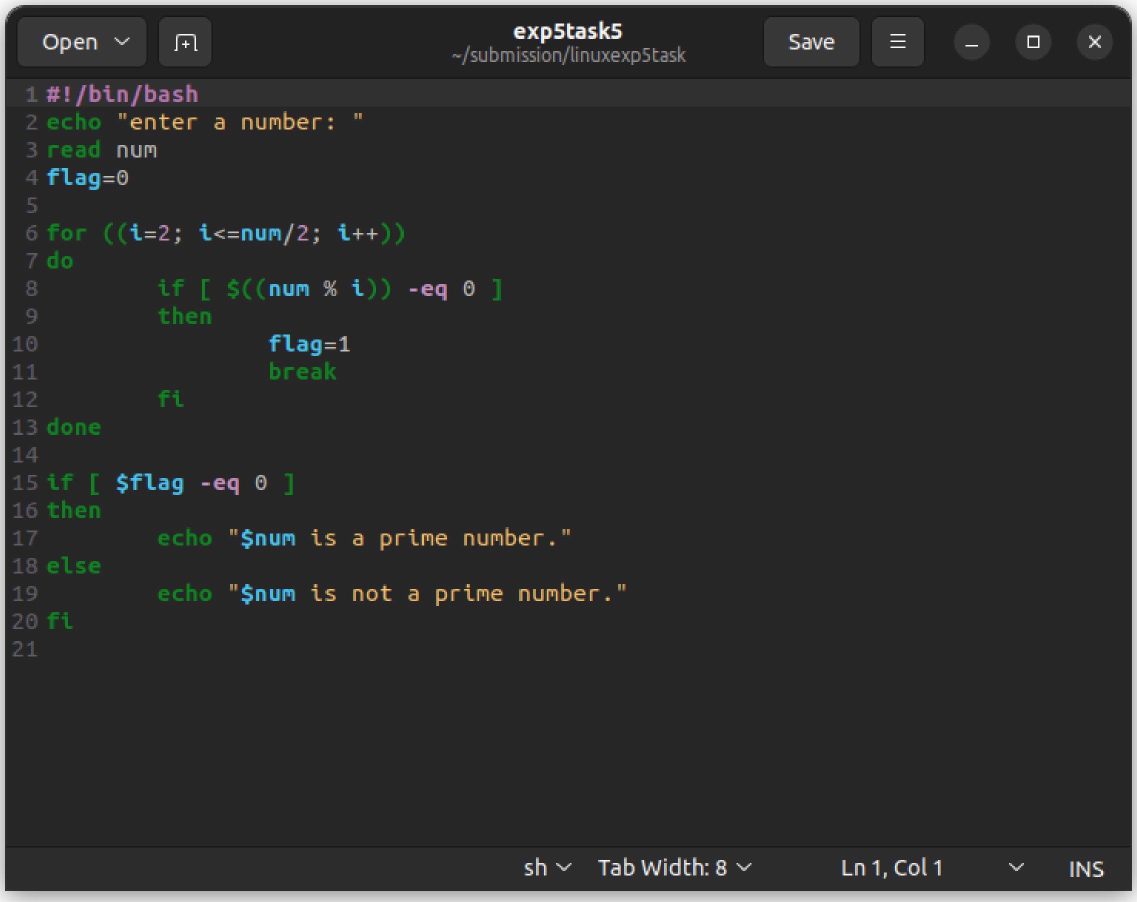


```
pavani@UBUNTU:~/Documents/linuxexp5task$ ./exp5task1
Hello, World!
pavani@UBUNTU:~/Documents/linuxexp5task$ ./exp5task2
bash: ./exp5task2: Permission denied
pavani@UBUNTU:~/Documents/linuxexp5task$ chmod +x exp5task2
pavani@UBUNTU:~/Documents/linuxexp5task$ ./exp5task2
Enter your name:
pavani
Hello, pavani! Welcome to Shell Scripting.
pavani@UBUNTU:~/Documents/linuxexp5task$ ./exp5task3
bash: ./exp5task3: Permission denied
pavani@UBUNTU:~/Documents/linuxexp5task$ chmod +x exp5task3
pavani@UBUNTU:~/Documents/linuxexp5task$ ./exp5task3
Enter first number:
12
Enter second number:
4
Addition: 16
Subtraction: 8
Multiplication: 48
Division: 3
pavani@UBUNTU:~/Documents/linuxexp5task$ ./exp5task4
bash: ./exp5task4: Permission denied
pavani@UBUNTU:~/Documents/linuxexp5task$ chmod +x exp5task4
pavani@UBUNTU:~/Documents/linuxexp5task$ ./exp5task4
Enter your age
20
You are eligible to vote.
pavani@UBUNTU:~/Documents/linuxexp5task$ ./exp5task4
```

TASK 5 - **Prime Number Check**

- **Goal:** Input number → check if prime.
- **Hint:** Loop from 2 to num/2, check divisibility using %.
- **Test Cases:**
  - Input: 7 → *Prime*
  - Input: 10 → *Not Prime*

COMMAND



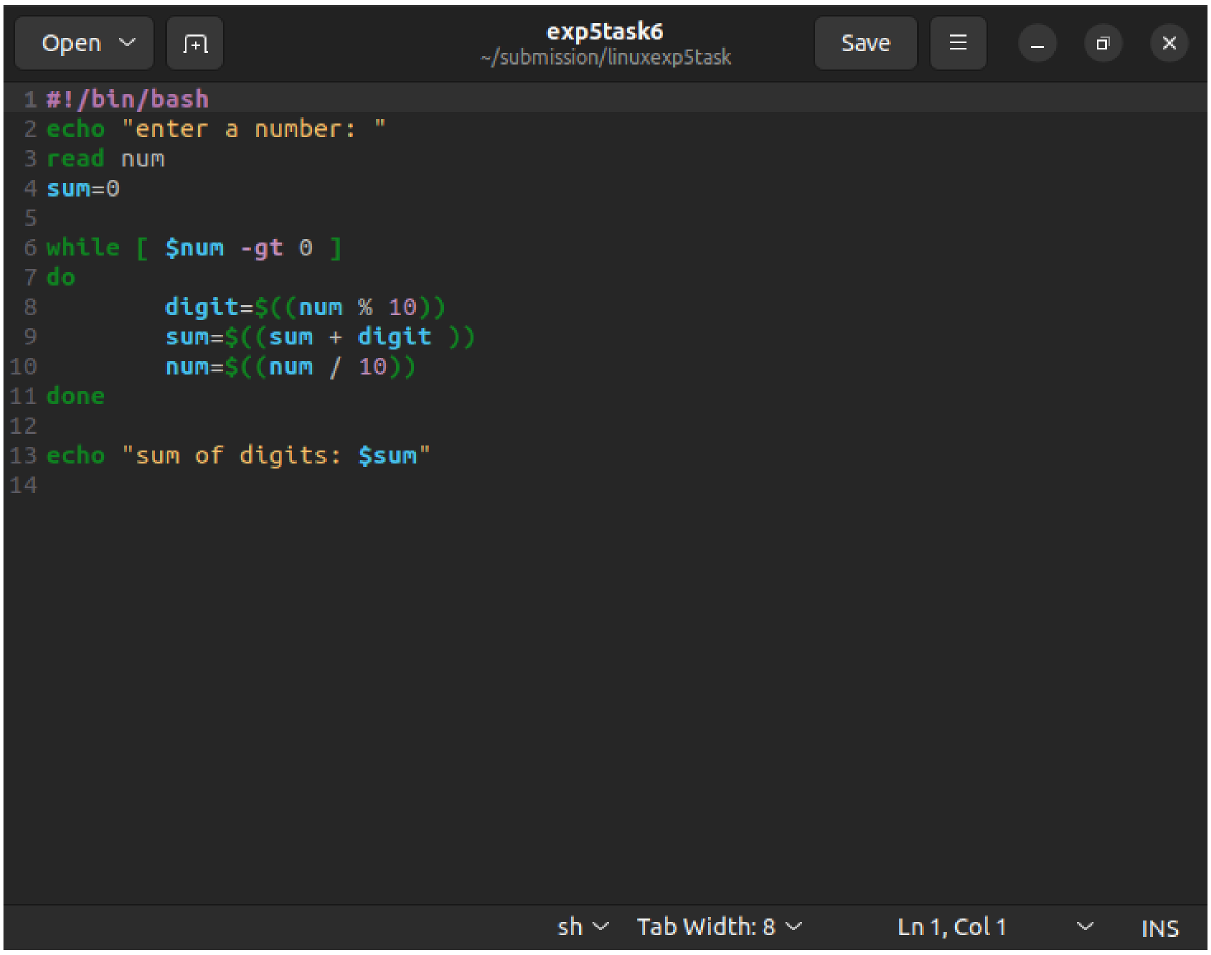
```
#!/bin/bash
echo "enter a number: "
read num
flag=0

for ((i=2; i<=num/2; i++))
do
        if [ $((num % i)) -eq 0 ]
        then
                flag=1
                break
        fi
done

if [ $flag -eq 0 ]
then
        echo "$num is a prime number."
else
        echo "$num is not a prime number."
fi

```

sh Tab Width: 8 Ln 1, Col 1 INS

TASK 6 - **Sum of Digits**

- **Goal:** Input a number, calculate sum of digits.
- **Hint:** Use % 10 to extract last digit, / 10 to reduce number.
- **Test Case:**
    - Input: 1234 → Output: 10

COMMAND

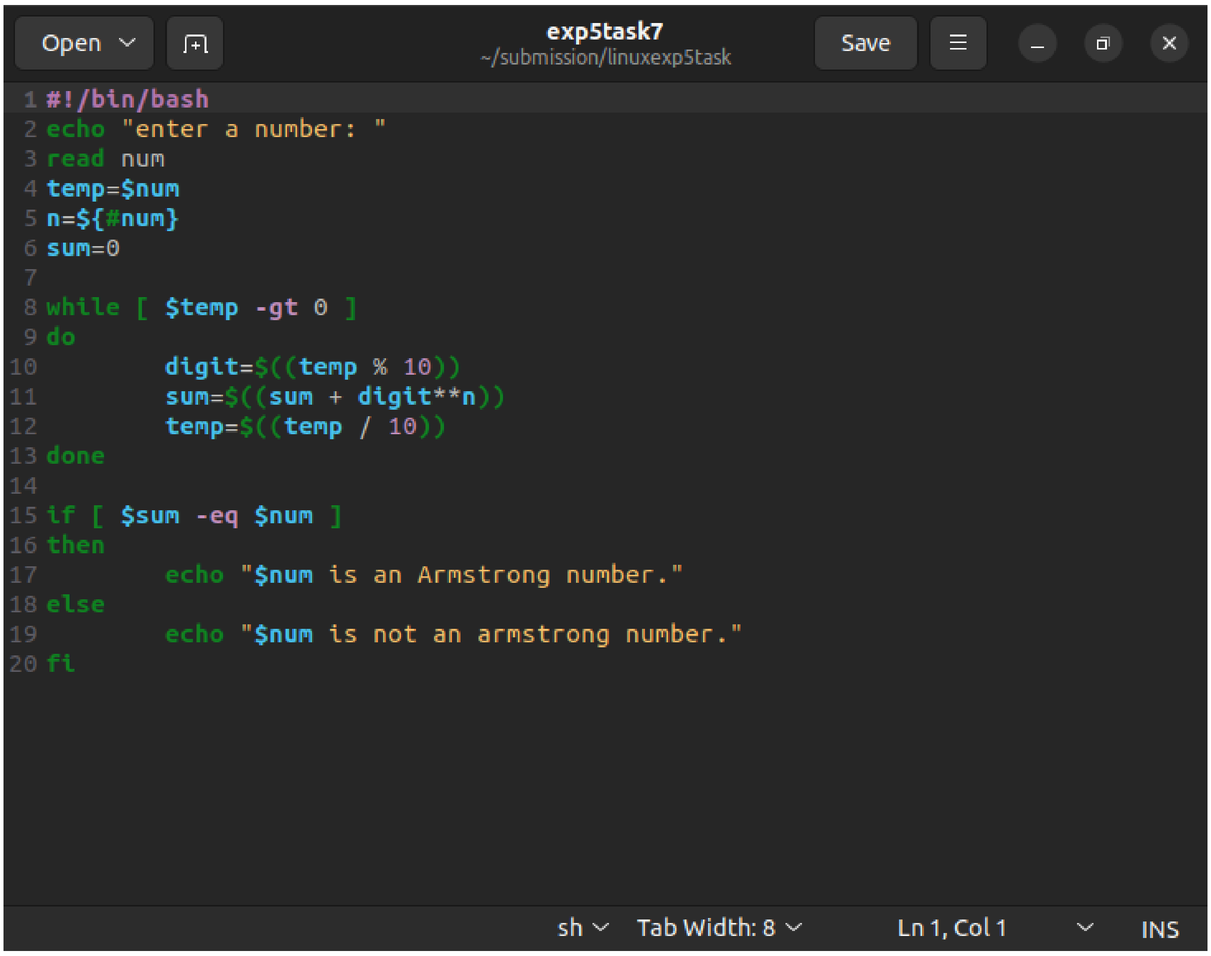


```
#!/bin/bash
echo "enter a number: "
read num
sum=0

while [ $num -gt 0 ]
do
        digit=$((num % 10))
        sum=$((sum + digit ))
        num=$((num / 10))
done

echo "sum of digits: $sum"

```

TASK 7 - **Number Check**

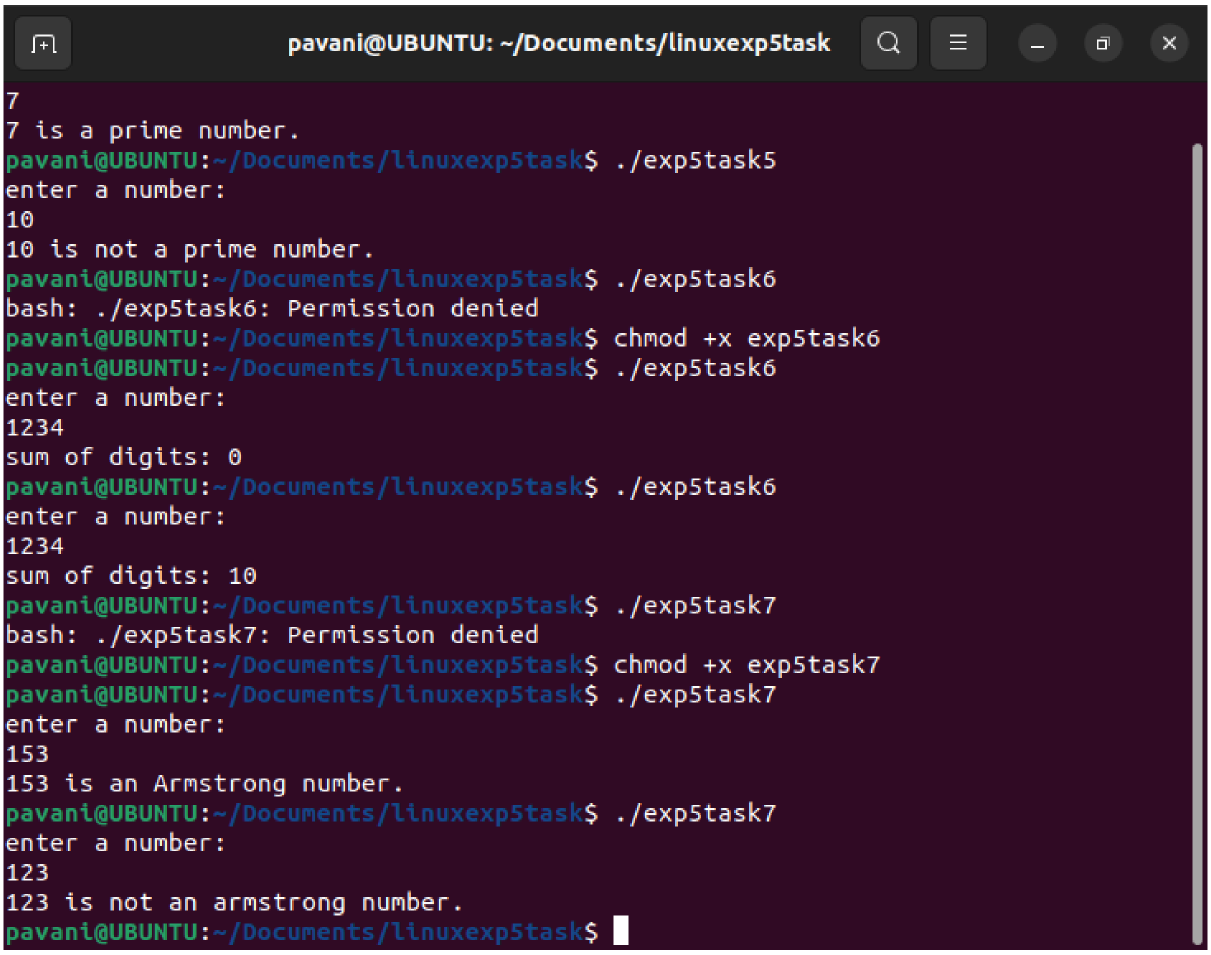
- **Goal:** Check if number is Armstrong.
- **Hint:** Use digit**n where n = number of digits (${#num}).
- **Test Cases:**
    - Input: 153 → Output: *Armstrong*
    - Input: 123 → Output: *Not Armstrong*

COMMAND



```
#!/bin/bash
echo "enter a number: "
read num
temp=$num
n=${#num}
sum=0

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

if [ $sum -eq $num ]
then
	echo "$num is an Armstrong number."
else
	echo "$num is not an armstrong number."
fi
```

OUTPUT OF TASK 5,6,7



```
7
7 is a prime number.
pavani@UBUNTU:~/Documents/linuxexp5task$ ./exp5task5
enter a number:
10
10 is not a prime number.
pavani@UBUNTU:~/Documents/linuxexp5task$ ./exp5task6
bash: ./exp5task6: Permission denied
pavani@UBUNTU:~/Documents/linuxexp5task$ chmod +x exp5task6
pavani@UBUNTU:~/Documents/linuxexp5task$ ./exp5task6
enter a number:
1234
sum of digits: 0
pavani@UBUNTU:~/Documents/linuxexp5task$ ./exp5task6
enter a number:
1234
sum of digits: 10
pavani@UBUNTU:~/Documents/linuxexp5task$ ./exp5task7
bash: ./exp5task7: Permission denied
pavani@UBUNTU:~/Documents/linuxexp5task$ chmod +x exp5task7
pavani@UBUNTU:~/Documents/linuxexp5task$ ./exp5task7
enter a number:
153
153 is an Armstrong number.
pavani@UBUNTU:~/Documents/linuxexp5task$ ./exp5task7
enter a number:
123
123 is not an armstrong number.
pavani@UBUNTU:~/Documents/linuxexp5task$
```

Observation

The script correctly displayed different command line arguments when given during execution.

Arrays helped in storing multiple arguments together and allowed easy access to them.

Conditional statements helped the script to take different actions based on the input values.

For example, when enough arguments were not provided, the condition displayed a suitable message.

Conclusion

Command line arguments allow the user to pass input values directly while executing a script, making it more dynamic.

Arrays in Bash provide a way to handle multiple values efficiently.

Conditional statements give logical control to the script, allowing it to make decisions

This experiment successfully demonstrated how command line arguments, arrays, and conditional statements are used in shell programming.