

# Experiment 6: Shell Programming

## Aim

The aim of this experiment is to understand and implement advanced concepts in shell programming such as loops, loop control, input/output redirection, shell functions, use of regular expressions, and debugging scripts.

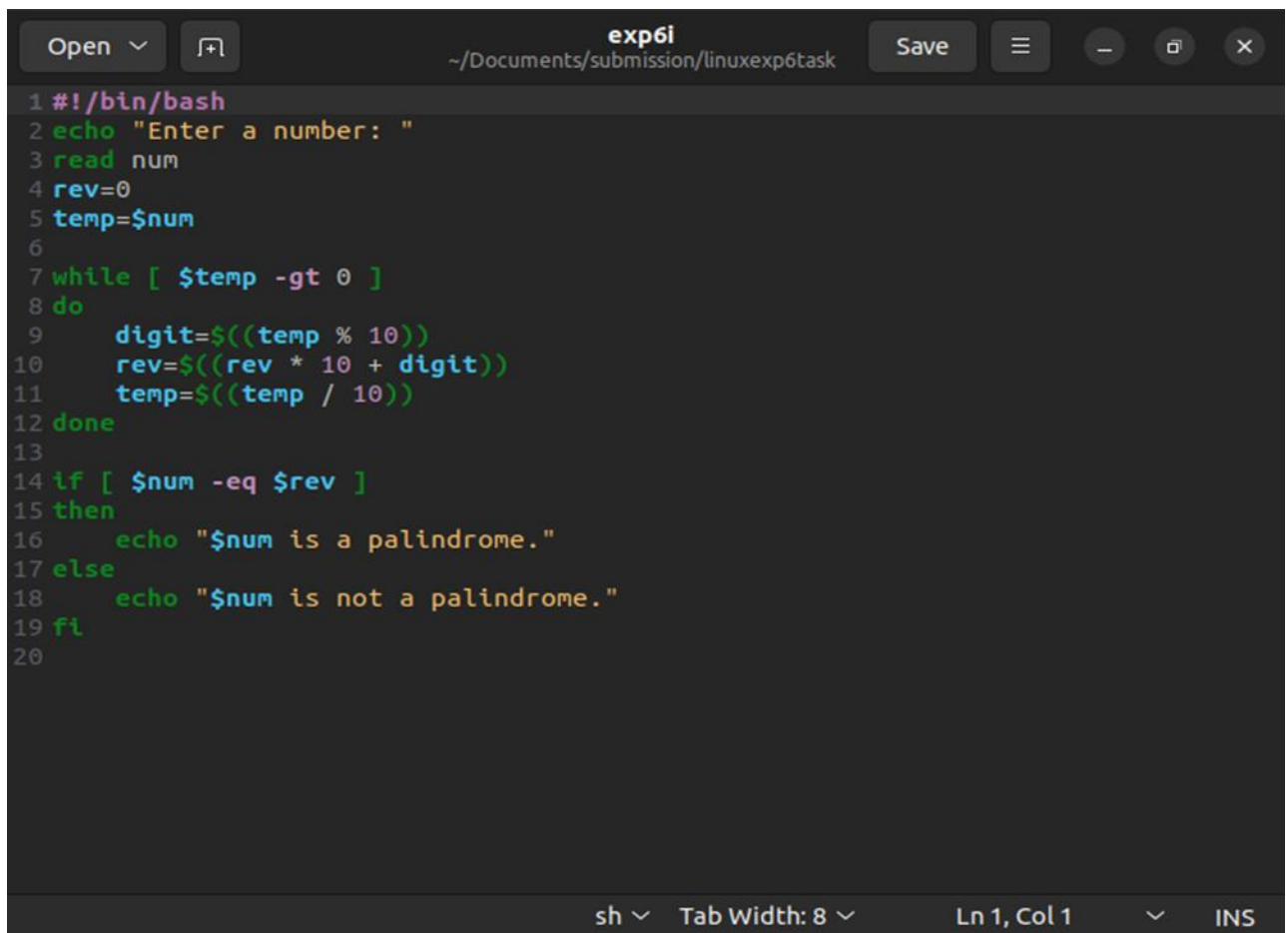
## Requirements

1. Linux operating system (Ubuntu, Fedora, etc.)
2. Access to a terminal
3. Text editor (nano, gedit, or vi) to write scripts
4. Basic knowledge of shell programming
5. Bash shell installed

## Lab Exercises

- i. Pallindrome check
- ii. GCD & LCM
- iii. Sorting Numbers






## Commands:





```
1 #!/bin/bash
2 echo "Enter a number: "
3 read num
4 rev=0
5 temp=$num
6
7 while [ $temp -gt 0 ]
8 do
9     digit=$((temp % 10))
10    rev=$((rev * 10 + digit))
11    temp=$((temp / 10))
12 done
13
14 if [ $num -eq $rev ]
15 then
16     echo "$num is a palindrome."
17 else
18     echo "$num is not a palindrome."
19 fi
20
```






The image shows a terminal window with a dark background. The window title is "exp6i" and the path is "~/Documents/submission/linuxexp6task". The script is a bash program that prompts the user to enter a number, reads it, and then checks if it is a palindrome by reversing the digits and comparing the original number with the reversed one. The script uses a while loop to extract digits and a do-while loop to build the reversed number. It then uses an if statement to output the result.

sh ▾ Tab Width: 8 ▾ Ln 1, Col 1 ▾ INS




Open  exp6ii  
~/Documents/submission/linuxexp6task Save    

```
1 #!/bin/bash
2 echo "Enter two numbers: "
3 read a b
4
5 x=$a
6 y=$b
7 while [ $y -ne 0 ]
8 do
9     temp=$y
10    y=$((x % y))
11    x=$temp
12 done
13 gcd=$x
14
15 lcm=$(( (a * b) / gcd ))
16
17 echo "GCD: $gcd"
18 echo "LCM: $lcm"
19
```

sh  Tab Width: 8  Ln 1, Col 1  INS

Open  exp6iii  
~/Documents/submission/linuxexp6task Save    

```
1 #!/bin/bash
2 echo "Enter numbers separated by space: "
3 read -a arr
4
5 echo "Ascending Order: "
6 printf "%s\n" "${arr[@]}" | sort -n
7
8 echo "Descending Order: "
9 printf "%s\n" "${arr[@]}" | sort -nr
10
```

sh  Tab Width: 8  Ln 1, Col 1  INS

## OUTPUTS

```
pavani@UBUNTU: ~/Documents/submission/linuxexp6task
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ vim exp6i
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ chmod +x exp6i
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ ./exp6i
Enter a number:
101
101 is a palindrome.
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ vim exp6ii
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ ./exp6ii
bash: ./exp6ii: Permission denied
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ chmod +x exp6ii
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ ./exp6ii
Enter two numbers:
21 42
GCD: 21
LCM: 42
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ ./exp6ii
Enter two numbers:
23 70
GCD: 1
LCM: 1610
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ vim exp6iii
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ chmod +x exp6iii
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ ./exp6iii
Enter numbers separated by space:
2 33 1 77 44 9
Ascending Order:
1
2
9
33
44
77
Descending Order:
77
44
33
9
2
1
pavani@UBUNTU:~/Documents/submission/linuxexp6task$
```

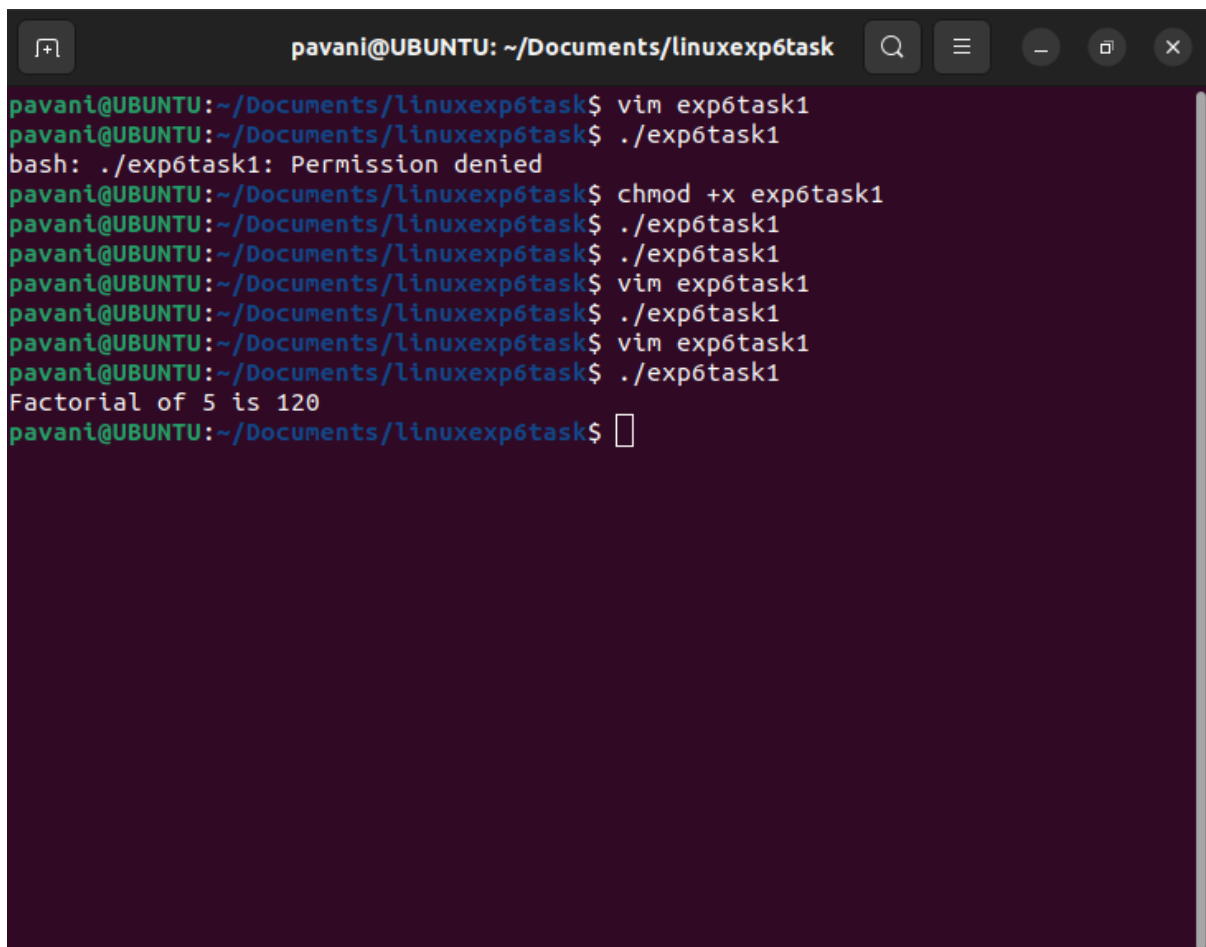
# ASSIGNMENT

## TASK 1:

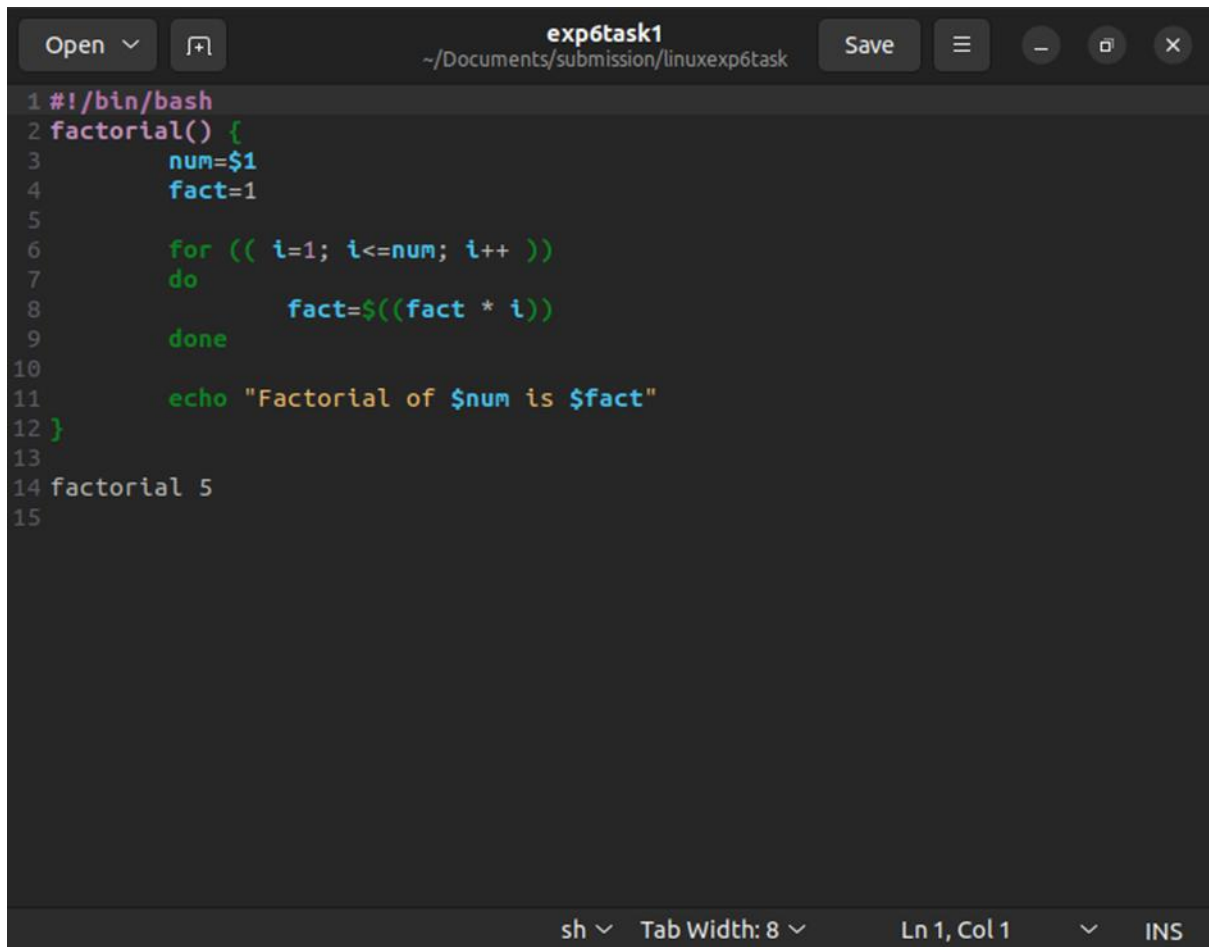
Write a function to calculate the factorial of a number using a loop.

## EXPLANATION

Factorial means multiplying numbers from  $n$  down to 1. For example,  $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$ . We can do this using a for or while loop.

A terminal window titled 'pavani@UBUNTU: ~/Documents/linuxexp6task' with standard Ubuntu window controls. The terminal shows a series of commands and their outputs. The user creates a file 'exp6task1' with 'vim', attempts to run it, gets a 'Permission denied' error, then uses 'chmod +x exp6task1' to make it executable. After running it again, the output is 'Factorial of 5 is 120'. The user then runs 'vim exp6task1' twice more and runs the script again.

```
pavani@UBUNTU:~/Documents/linuxexp6task$ vim exp6task1
pavani@UBUNTU:~/Documents/linuxexp6task$ ./exp6task1
bash: ./exp6task1: Permission denied
pavani@UBUNTU:~/Documents/linuxexp6task$ chmod +x exp6task1
pavani@UBUNTU:~/Documents/linuxexp6task$ ./exp6task1
pavani@UBUNTU:~/Documents/linuxexp6task$ ./exp6task1
pavani@UBUNTU:~/Documents/linuxexp6task$ vim exp6task1
pavani@UBUNTU:~/Documents/linuxexp6task$ ./exp6task1
pavani@UBUNTU:~/Documents/linuxexp6task$ vim exp6task1
pavani@UBUNTU:~/Documents/linuxexp6task$ ./exp6task1
Factorial of 5 is 120
pavani@UBUNTU:~/Documents/linuxexp6task$
```

A screenshot of a terminal window titled "exp6task1" with the path "~/Documents/submission/linuxexp6task". The window contains a bash script for calculating a factorial. The script starts with a shebang, defines a function "factorial()" that takes a number "num" and calculates its factorial using a "for" loop, and then calls the function with the argument "5".

```
1 #!/bin/bash
2 factorial() {
3     num=$1
4     fact=1
5
6     for (( i=1; i<=num; i++ ))
7     do
8         fact=$((fact * i))
9     done
10
11     echo "Factorial of $num is $fact"
12 }
13
14 factorial 5
15
```

## TASK 2:

Write a script that reads a filename and counts how many times a given word appears in it.

### EXPLANATION

Explanation:

We will open a file using `fopen`, read words using `fscanf`, and compare each word with the target word using `strcmp`. OO

```
exp6task2
~/Documents/submission/linuxexp6task
Open Save
1 #!/bin/bash
2
3 read -p "Enter filename: " filename
4 if [ ! -f "$filename" ]; then
5     echo "File does not exist."
6     exit 1
7 fi
8
9 read -p "Enter word to count: " word
10 count=$(grep -o -i "$word" "$filename" | wc -l)
11 echo "The word '$word' appears $count times in $filename."
12
13
sh Tab Width: 8 Ln 1, Col 1 INS
```

OUR

```
pavani@UBUNTU: ~/Documents/submission/linuxexp6task
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ vim exp6task2
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ chmod +x exp6task2
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ ./exp6task2
Enter filename: submission
File does not exist.
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ ./exp6task2
Enter filename: cron_log.txt
File does not exist.
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ ./exp6task2
Enter filename: exp6task2
Enter word to count: echo
The word 'echo' appears 2 times in exp6task2.
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ vim exp6task3
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ chmod +x exp6task3
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ ./exp6task3
Enter the value of N:
7
Fibonacci sequence up to 7 terms:
0 1 1 2 3 5 8
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ vim exp6task4
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ chmod +x exp6task4
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ ./exp6task4
Enter an email address:
happy@yahoo.com
Valid email address
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ vim
```

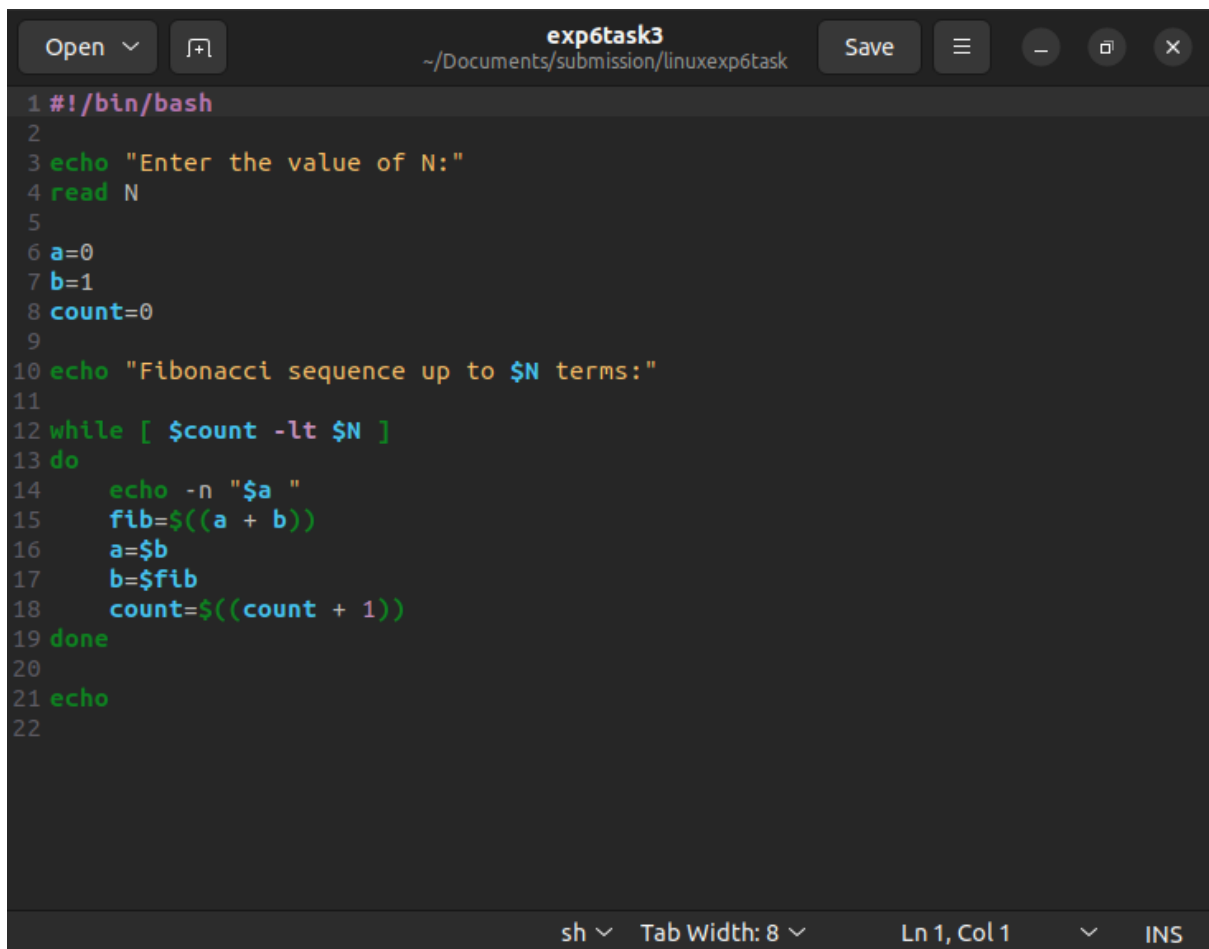
### TASK 3;

Write a script that generates the first N Fibonacci numbers using a while loop.

EXPLANATION: fibonacci series starts with 0, 1 and then each number is the sum of the previous two. Example: 0, 1, 1, 2, 3, 5, 8....



## COMMAND



```
1 #!/bin/bash
2
3 echo "Enter the value of N:"
4 read N
5
6 a=0
7 b=1
8 count=0
9
10 echo "Fibonacci sequence up to $N terms:"
11
12 while [ $count -lt $N ]
13 do
14     echo -n "$a "
15     fib=$((a + b))
16     a=$b
17     b=$fib
18     count=$((count + 1))
19 done
20
21 echo
22
```

OUTPUT in above image

### TASK 4:

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

```

1 #!/bin/bash
2
3 echo "Enter an email address:"
4 read email
5
6 # Regex for email validation
7 pattern="^[a-zA-Z0-9._%+-]+\@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$"
8
9 if [[ $email =~ ^[a-zA-Z0-9._%+-]+\@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$ ]]; then
10     echo "Valid email address"
11 else
12     echo "Invalid email address"
13 fi
14

```

Loading file "/home/pavani/Documents/sub... sh Tab Width: 8 Ln 1, Col 1 INS

OUTPUT in above image

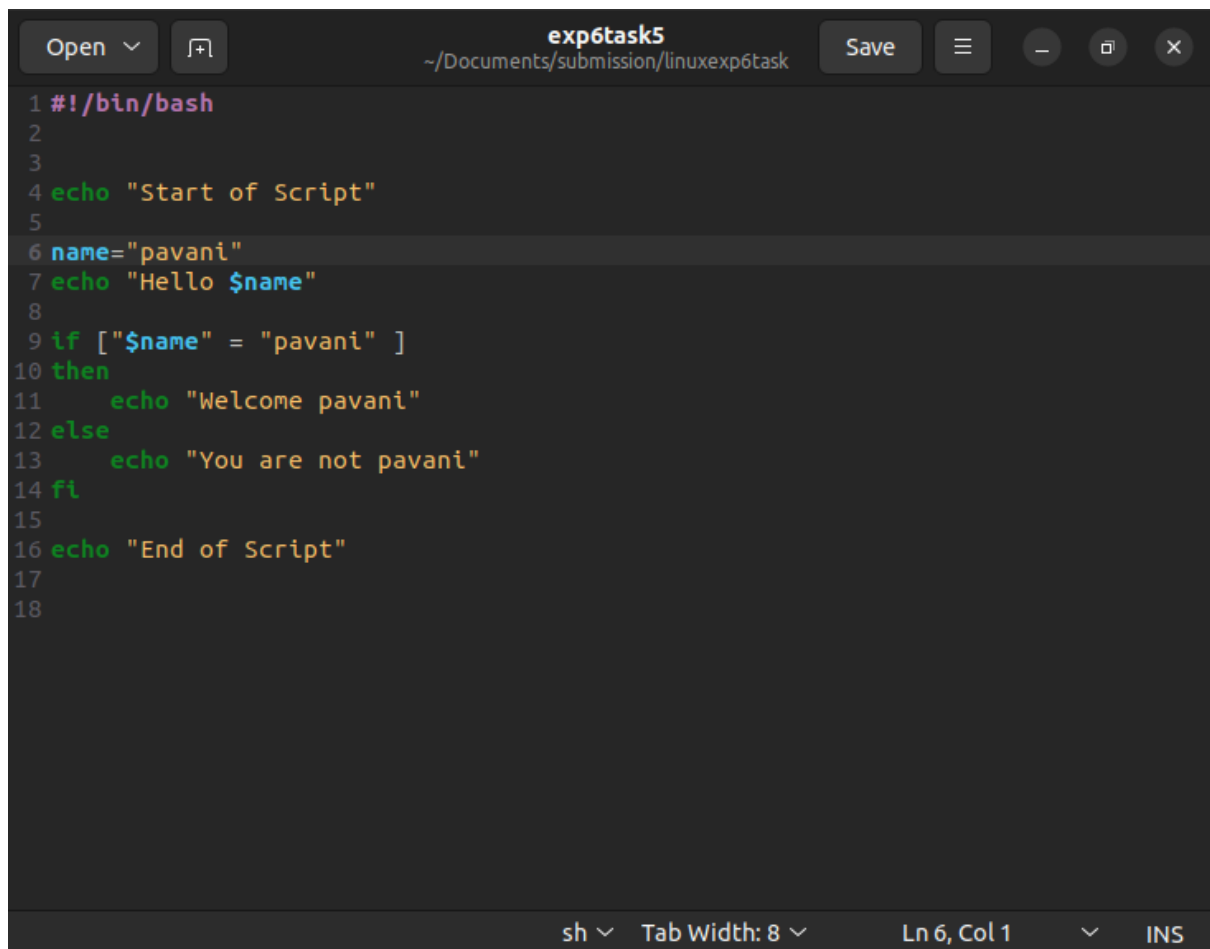
## TASK 5:

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

## EXPLANATION

Actually bash -x is used for debugging shell scripts, not C directly. But we can create a shell script that compiles a C file with an error. Debug output

will show step-by-step execution.



The image shows a terminal window titled "exp6task5" with a dark background. The window has a menu bar with "Open", "Save", and window control buttons. The file path is "~/Documents/submission/linuxexp6task". The terminal contains a shell script with the following content:

```
1 #!/bin/bash
2
3
4 echo "Start of Script"
5
6 name="pavani"
7 echo "Hello $name"
8
9 if [ "$name" = "pavani" ]
10 then
11     echo "Welcome pavani"
12 else
13     echo "You are not pavani"
14 fi
15
16 echo "End of Script"
17
18
```

The status bar at the bottom shows "sh", "Tab Width: 8", "Ln 6, Col 1", and "INS".

```
pavani@UBUNTU: ~/Documents/submission/linuxexp6task
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ vim exp6task5
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ chmod +x exp6task5
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ ./exp6task5
Start of Script
./exp6task5: line 16: unexpected EOF while looking for matching `"'
./exp6task5: line 18: syntax error: unexpected end of file
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ bash exp6task5
Start of Script
exp6task5: line 16: unexpected EOF while looking for matching `"'
exp6task5: line 18: syntax error: unexpected end of file
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ vim exp6task5
pavani@UBUNTU:~/Documents/submission/linuxexp6task$ bash -x exp6task5
+ echo 'Start of Script'
Start of Script
+ name=pavani
+ echo 'Hello pavani'
Hello pavani
+ '[pavani' = pavani ']'
exp6task5: line 9: [pavani: command not found
+ echo 'You are not pavani'
You are not pavani
+ echo 'End of Script'
End of Script
pavani@UBUNTU:~/Documents/submission/linuxexp6task$
```

## Observation

Loops in shell help to repeat commands.

Loop control statements like break and continue control execution flow.

I/O redirection is powerful for handling files and error messages.

Functions allow code reusability.

Regular expressions help in text searching and pattern matching.

Debugging tools like bash -x help in finding errors in scripts.

## Conclusion

From this experiment, we conclude that advanced shell programming concepts make scripts more powerful and flexible. Loops and loop control help manage repeated tasks,

I/O redirection manages inputs and outputs, functions improve modularity, and regular expressions enhance text handling. Debugging techniques are essential for error-free execution of scripts.