Vidyalankar Polytechnic	DEPARTMENT OF COMPUTER ENGINEERING

Subject: Operating System	Subject Code:22516
Semester:5th Semester	Course: Computer Engineering
Laboratory No: V118	Name of Subject Teacher: Prof. Natasha Brahme
Name of Student: Siddharth Shah	Roll Id: 22203A0041

Experiment No:	11
Title of Experiment	Execute shell script by using for statement

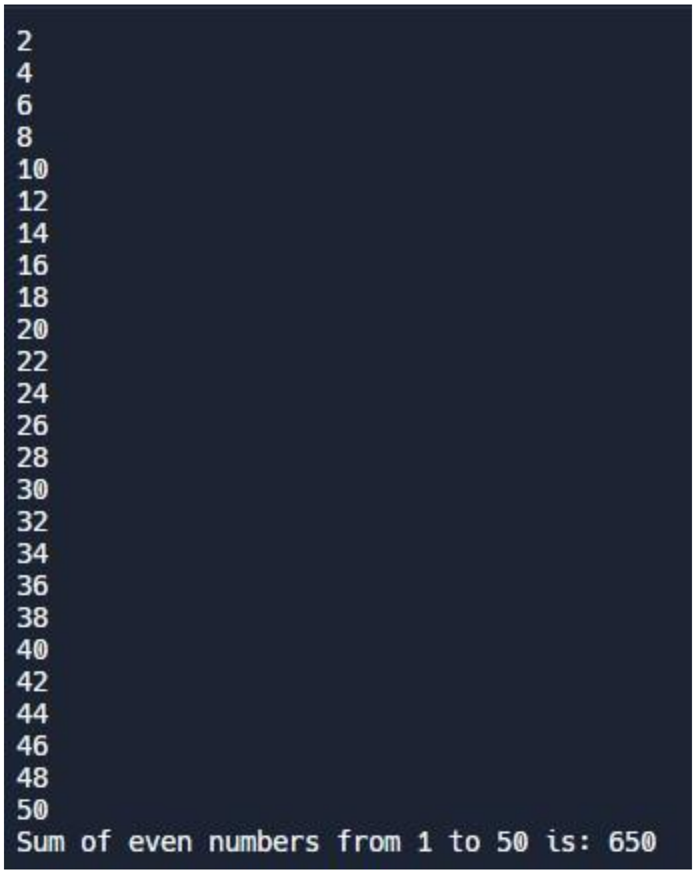
SET QUESTIONS

1) **Write a shell script to print even number from 1 to 50 and sum of them**

➔

```
#!/bin/bash
sum=0
for ((i=1; i<=50; i++))
do
  if ((i % 2 == 0));
  then
    echo $i  # Print the even number
    sum=$((sum + i))
  fi
done
echo "Sum of even numbers from 1 to 50 is: $sum"
```

Output:-



```
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
50
Sum of even numbers from 1 to 50 is: 650
```

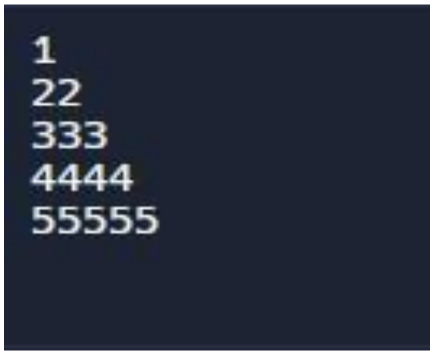
2) **Write shell script to print following output**

1
22
333
4444
5555

➔

```
#!/bin/bash
for ((i=1; i<=5; i++))
do
  for ((j=1; j<=i; j++))
  do
    echo -n "$i"
  done
  echo
done
```

Output:-



```
1
22
333
4444
55555
```

XII. PRACTRICAL RELATED QUESTIONS.

1) Give output of the following.

➔

a)

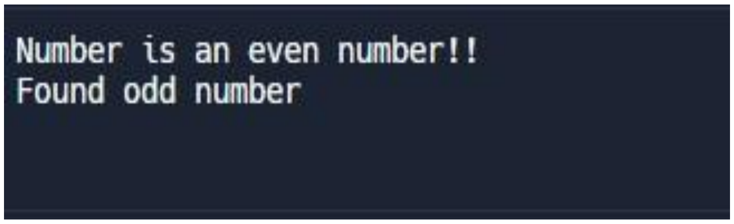
```
#!/bin/sh
NUMBERS="123456 7"
for NUM in $NUMS
do
Q='expr $NUM % 2'
if [ $Q -eq 0]
then
echo "Number is an even number!!"
continue
fi
echo "Found odd number"
done
```

Ans:-

```
#!/bin/sh

NUMBERS="123456 7"

for NUM in $NUMBERS

do

Q=$(expr $NUM % 2)

if [ $Q -eq 0 ]

then

 echo "Number is an even number!!"

continue

fi
```

```
echo "Found odd number"

done
```

Output:-



```
Number is an even number!!
Found odd number
```

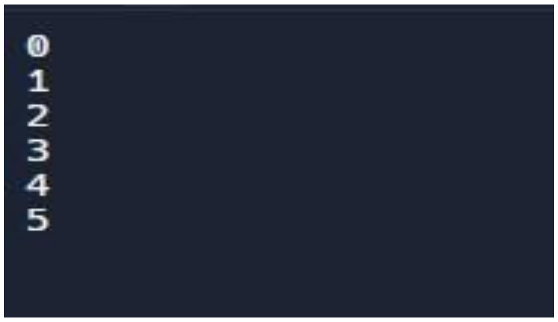
b)

```
#!/bin/sh
a=0
while ($a-lt 10]
do
echo $a
if [ $a -eq 5]
then
break
fi
a='expr $a + 1'
done
```

Ans:-

```
#!/bin/sh

a=0

while [ $a -lt 10 ]

do

 echo $a

if [ $a -eq 5 ]

 then

 break

fi

 a=$(expr $a + 1)

done
```

Output:-



```
0
1
2
3
4
5
```

2) **State the difference between iteration and recursion.**

➔

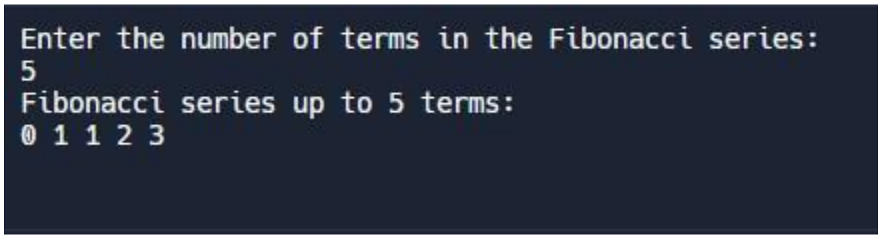
Property	**Recursion**	**Iteration**
Definition	Function calls itself.	A set of instructions repeatedly executed.
Application	For functions	For loops
Termination	Through base case, where there will be no function call	When the termination condition for the iterator ceases to be satisfied
Usage	Used when code size needs to be small and time complexity is not an issue	Used when time complexity needs to be balanced against an expanded code size
Code size	Smaller code size	Larger code size
Time Complexity	Very high (generally exponential) time complexity	Relatively lower time complexity (generally polynomial-logarithmic)

3) Write a shell script to display Fibonacci series for n numbers.

➔

```
#!/bin/bash
echo "Enter the number of terms in the Fibonacci series:"
read n
a=0
b=1
echo "Fibonacci series up to $n terms:"
for ((i=0; i<n; i++))
do
  echo -n "$a "
  fn=$((a + b))
  a=$b
  b=$fn
done
echo
```

Output:-



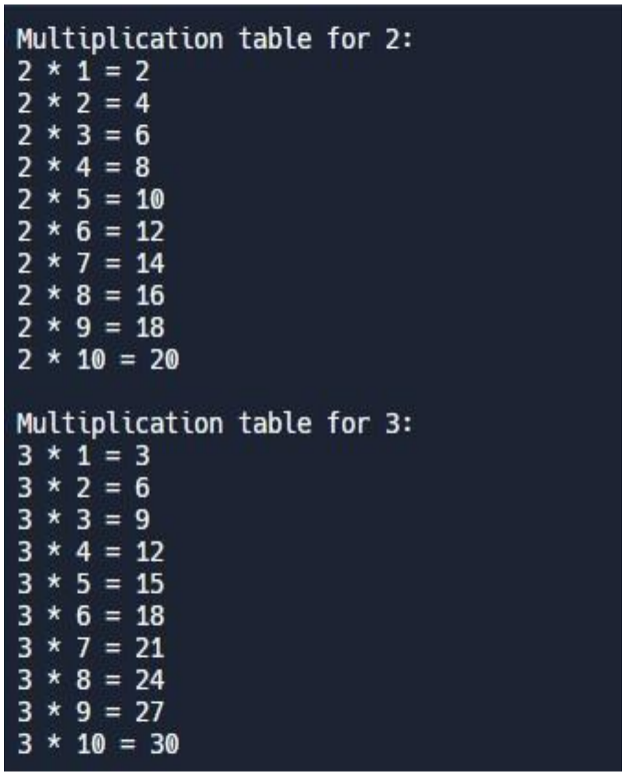
```
Enter the number of terms in the Fibonacci series:
5
Fibonacci series up to 5 terms:
0 1 1 2 3
```

4) Write a shell script to display tables of 2 to 10 numbers (Like 2 * 1 = 2).

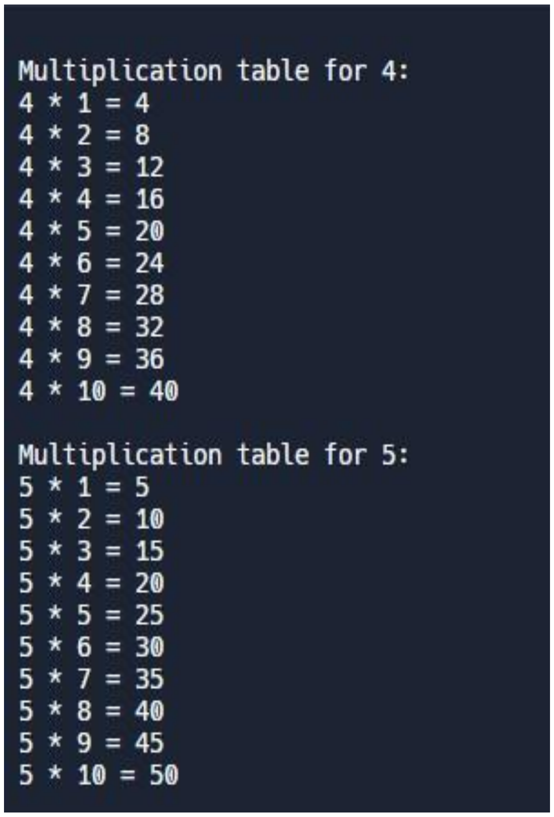
➔

```
#!/bin/bash
for ((i=2; i<=10; i++))
do
  echo "Multiplication table for $i:"
  for ((j=1; j<=10; j++))
  do
    result=$((i * j))
    echo "$i * $j = $result"
  done
  echo
done
```

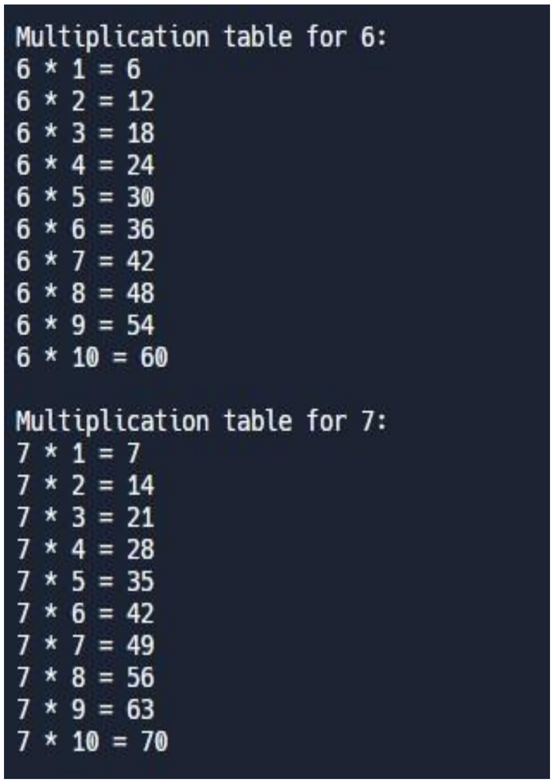
Output:-



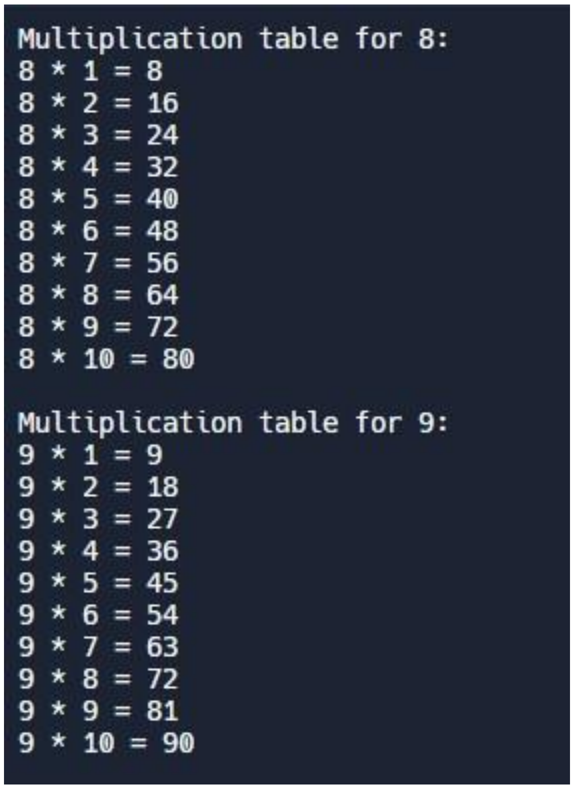
```
Multiplication table for 2:
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20

Multiplication table for 3:
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30
```



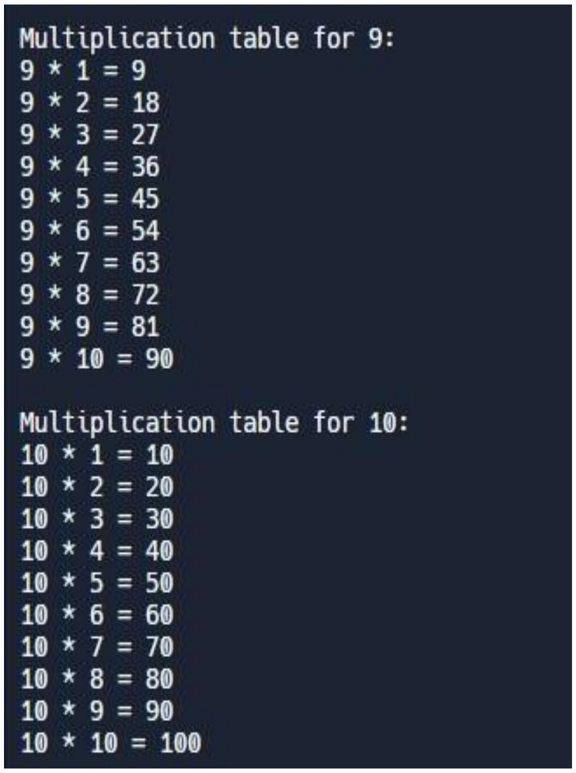
```
Multiplication table for 4:
4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
4 * 10 = 40

Multiplication table for 5:
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```



```
Multiplication table for 6:
6 * 1 = 6
6 * 2 = 12
6 * 3 = 18
6 * 4 = 24
6 * 5 = 30
6 * 6 = 36
6 * 7 = 42
6 * 8 = 48
6 * 9 = 54
6 * 10 = 60

Multiplication table for 7:
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
7 * 10 = 70
```



```
Multiplication table for 8:
8 * 1 = 8
8 * 2 = 16
8 * 3 = 24
8 * 4 = 32
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72
8 * 10 = 80

Multiplication table for 9:
9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
9 * 10 = 90
```



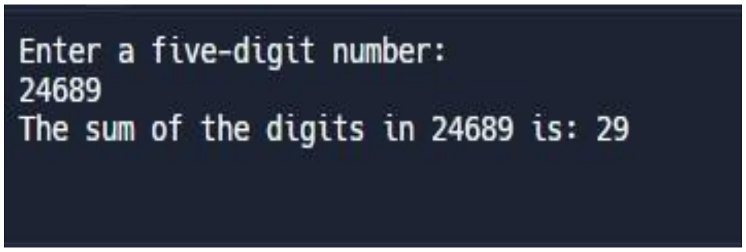
```
Multiplication table for 9:
9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
9 * 10 = 90

Multiplication table for 10:
10 * 1 = 10
10 * 2 = 20
10 * 3 = 30
10 * 4 = 40
10 * 5 = 50
10 * 6 = 60
10 * 7 = 70
10 * 8 = 80
10 * 9 = 90
10 * 10 = 100
```

5) **Write a shell script to accept five-digit number and perform addition of all digits.**

➔

```
#!/bin/bash
echo "Enter a five-digit number:"
read number
if [[ ! $number =~ ^[0-9]{5}$ ]];
then
  echo "Error: Please enter a valid five-digit number."
  exit 1
fi

sum=0
for (( i=0; i<5; i++ ))
do
  digit=${number:i:1}
  sum=$((sum + digit))
done
echo "The sum of the digits in $number is: $sum"
```

Output:-



```
Enter a five-digit number:
24689
The sum of the digits in 24689 is: 29
```

XIII. EXERCISE.

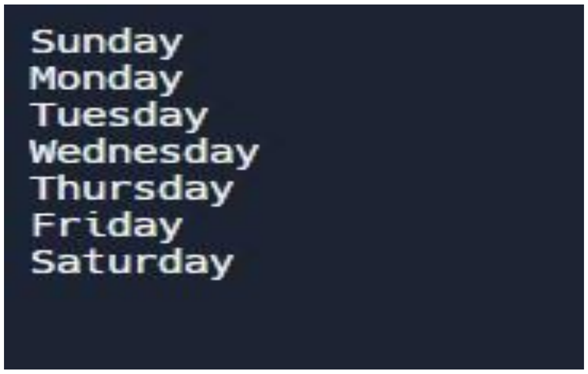
1) **Execute the script for the following.**

   i. The for loop using days of week.

➔

```
#!/bin/bash
days=("Sunday" "Monday" "Tuesday" "Wednesday" "Thursday" "Friday"
"Saturday")
for day in "${days[@]}"
do
  echo "$day"
done
```

Output:-



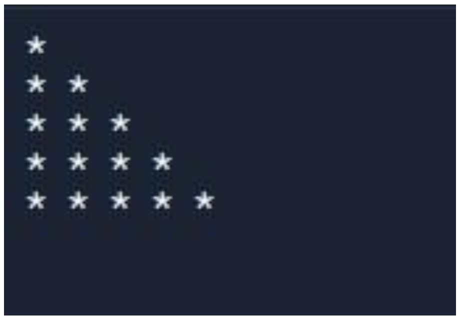
```
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
```

ii. The while loop to print different * patterns.

➔

```
#!/bin/bash
rows=5
i=1
while [ $i -le $rows ]
do
  for ((j=1; j<=i; j++))
  do
    echo -n "* "
  done
  echo
  i=$((i + 1))
done
```

Output:-



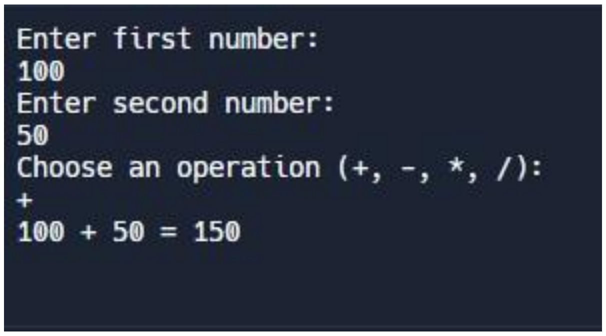
```
*
* *
* * *
* * * *
* * * * *
```

iii. The case statement for performing various mathematical operations

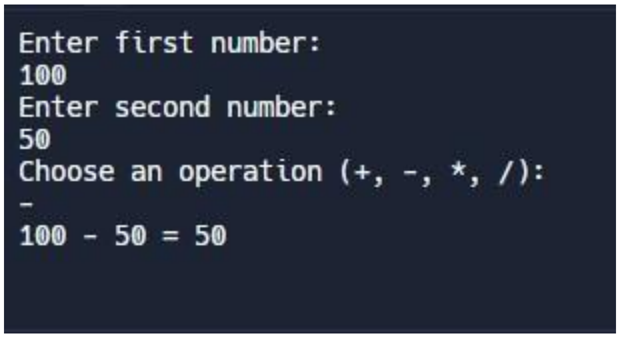
➔

```
#!/bin/bash
echo "Enter first number:"
read num1
echo "Enter second number:"
read num2
echo "Choose an operation (+, -, *, /):"
read operator
case $operator in
  +)
    result=$((num1 + num2))
    echo "$num1 + $num2 = $result"
    ;;
  -)
    result=$((num1 - num2))
    echo "$num1 - $num2 = $result"
    ;;
  \*)
    result=$((num1 * num2))
    echo "$num1 * $num2 = $result"
    ;;
  /)
    if [ $num2 -ne 0 ]; then
      result=$((num1 / num2))
      echo "$num1 / $num2 = $result"
    else
      echo "Error: Division by zero is not allowed!"
    fi
    ;;
  *)
    echo "Invalid operator!"
    ;;
esac
```

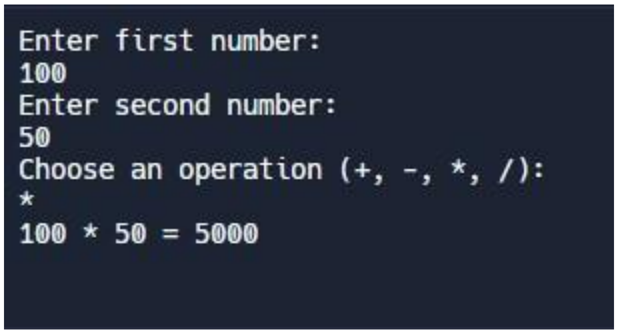
Output:-



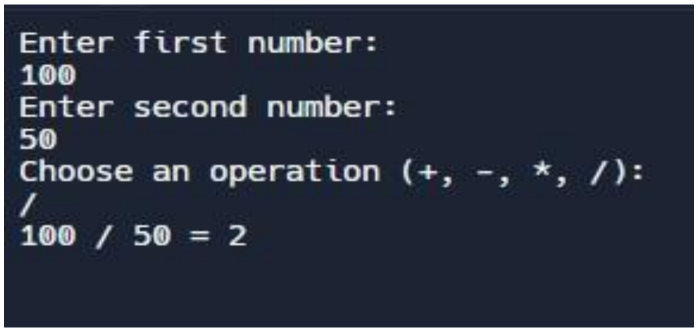
```
Enter first number:
100
Enter second number:
50
Choose an operation (+, -, *, /):
+
100 + 50 = 150
```



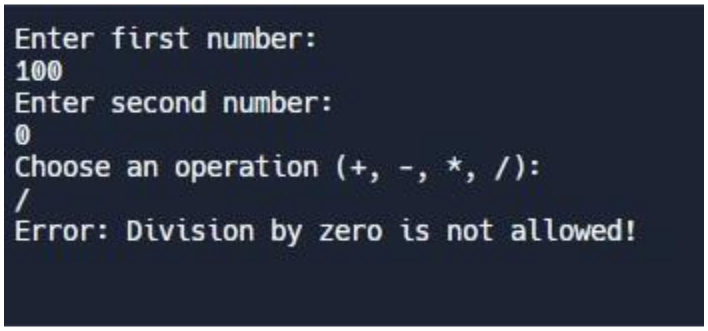
```
Enter first number:
100
Enter second number:
50
Choose an operation (+, -, *, /):
-
100 - 50 = 50
```



```
Enter first number:
100
Enter second number:
50
Choose an operation (+, -, *, /):
*
100 * 50 = 5000
```



```
Enter first number:
100
Enter second number:
50
Choose an operation (+, -, *, /):
/
100 / 50 = 2
```



```
Enter first number:
100
Enter second number:
0
Choose an operation (+, -, *, /):
/
Error: Division by zero is not allowed!
```

CONCLUSION:

We are able to do different programs using shell script and vi editor and also we successfully implemented shell script.