EmbedUR Systems Private Limited

# Bash Script Training

## Module -4

By:

A.Buvaneshkumaar

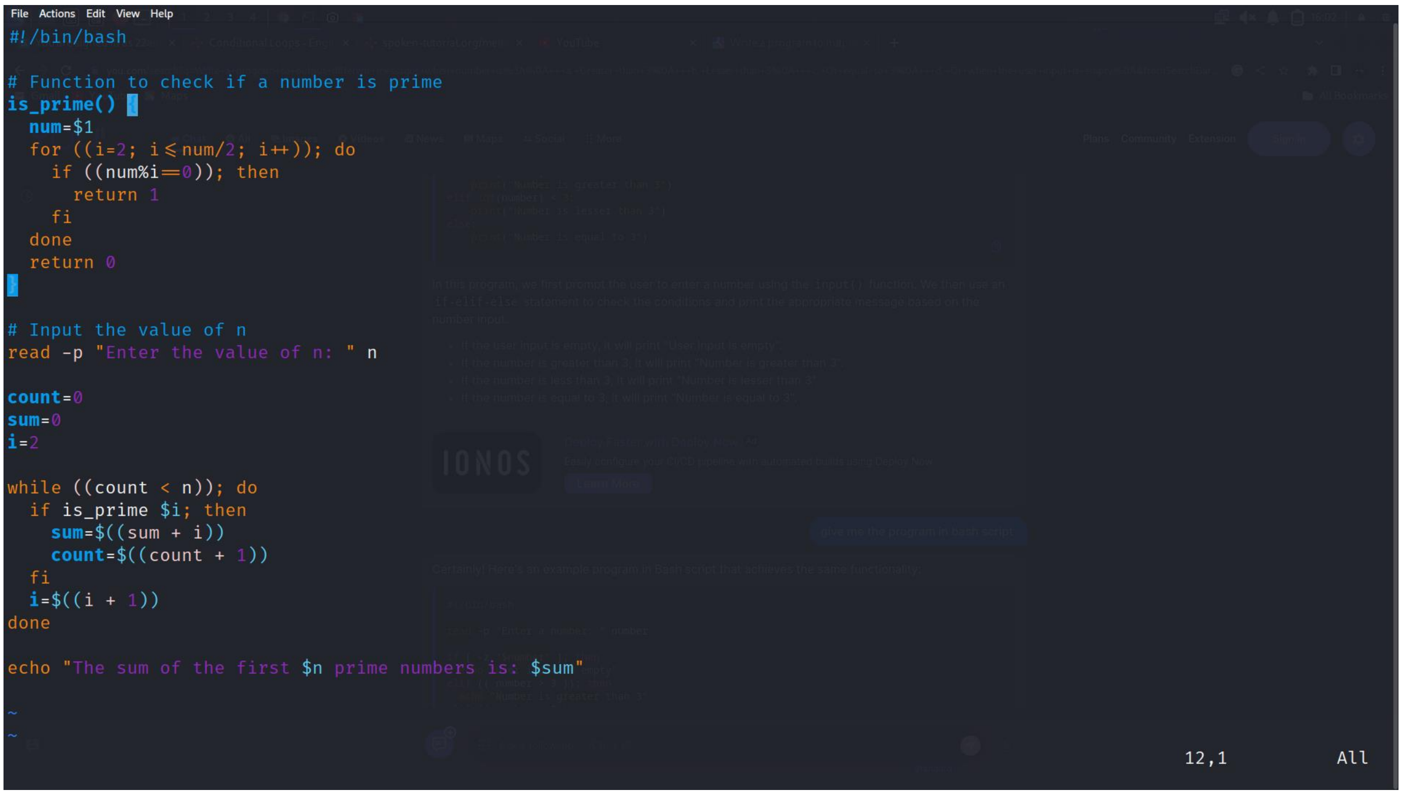
Mepco Schlenk Engineering College

Sivakasi , Virudhunagar District.,

## Conditional Loops

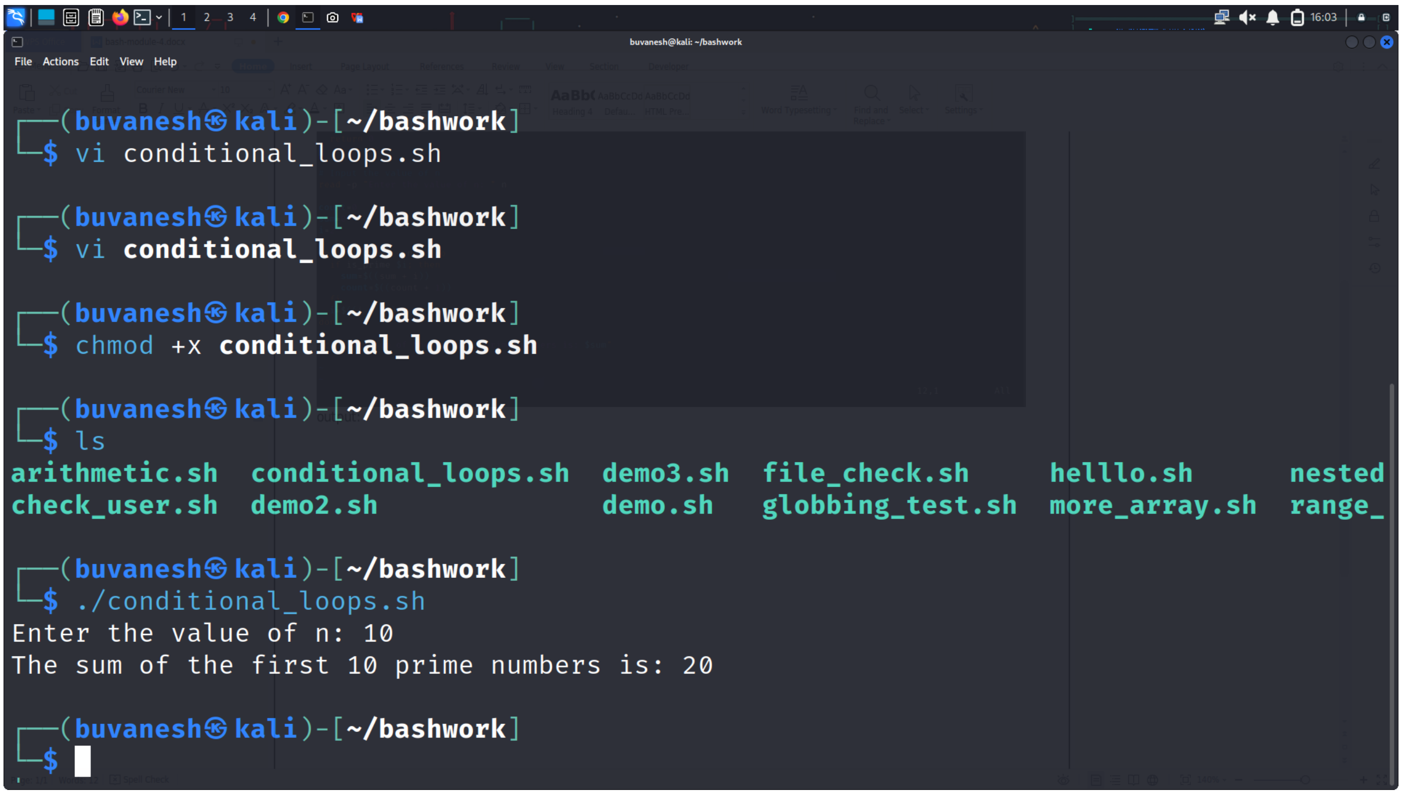
1) Find the sum of first n prime numbers.

**script:**



```
#!/bin/bash

# Function to check if a number is prime
is_prime() {
  num=$1
  for ((i=2; i<=num/2; i++)); do
    if ((num%i==0)); then
      return 1
    fi
  done
  return 0
}

# Input the value of n
read -p "Enter the value of n: " n

count=0
sum=0
i=2

while ((count < n)); do
  if is_prime $i; then
    sum=$((sum + i))
    count=$((count + 1))
  fi
  i=$((i + 1))
done

echo "The sum of the first $n prime numbers is: $sum"
~
~
                                                     12,1          All
```

**output:**

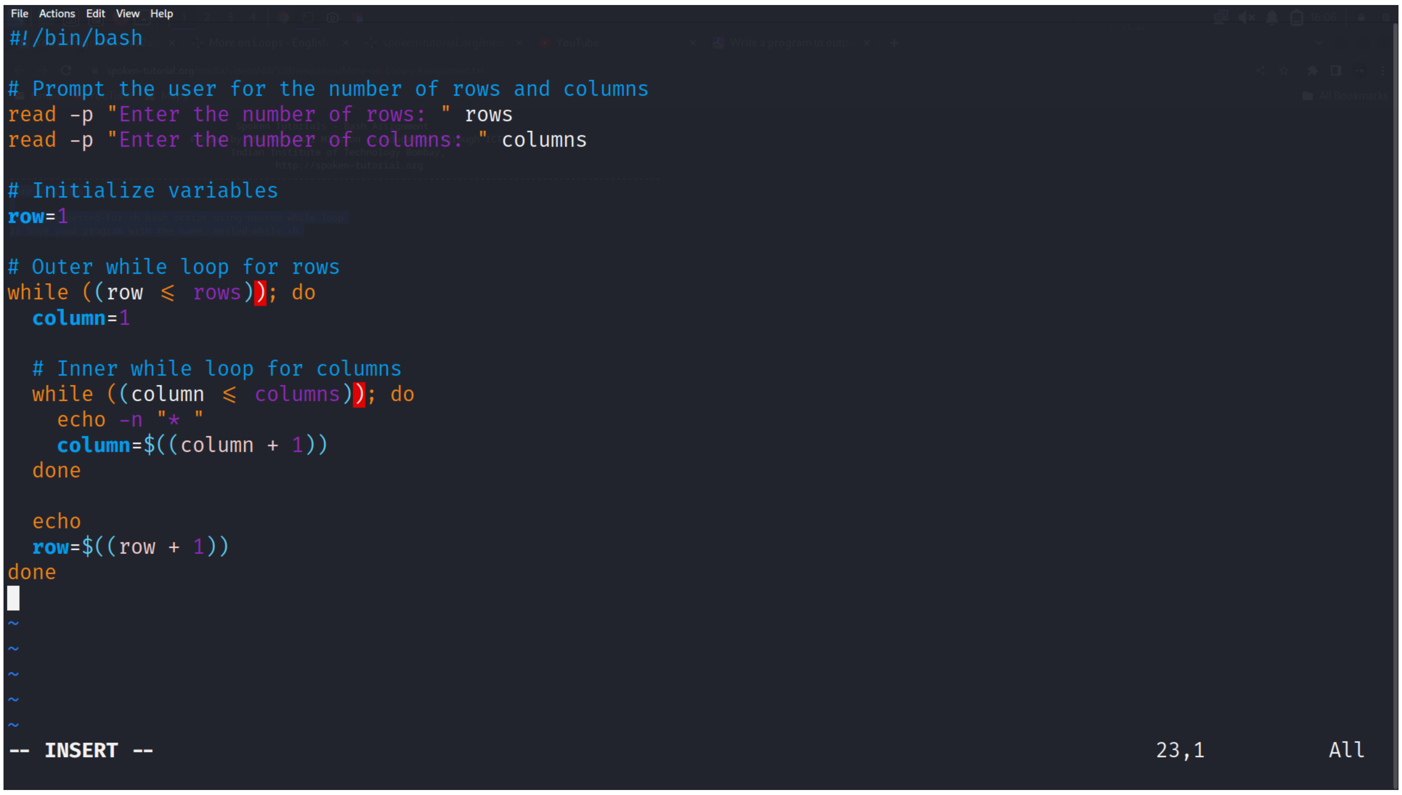


```
┌──(buvanesh㉿kali)-[~/bashwork]
└─$ vi conditional_loops.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ vi conditional_loops.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ chmod +x conditional_loops.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ ls
arithmetic.sh   conditional_loops.sh   demo3.sh   file_check.sh      helllo.sh       nested
check_user.sh   demo2.sh               demo.sh    globbing_test.sh   more_array.sh   range_

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ ./conditional_loops.sh
Enter the value of n: 10
The sum of the first 10 prime numbers is: 20

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ 
```

## More on Loops

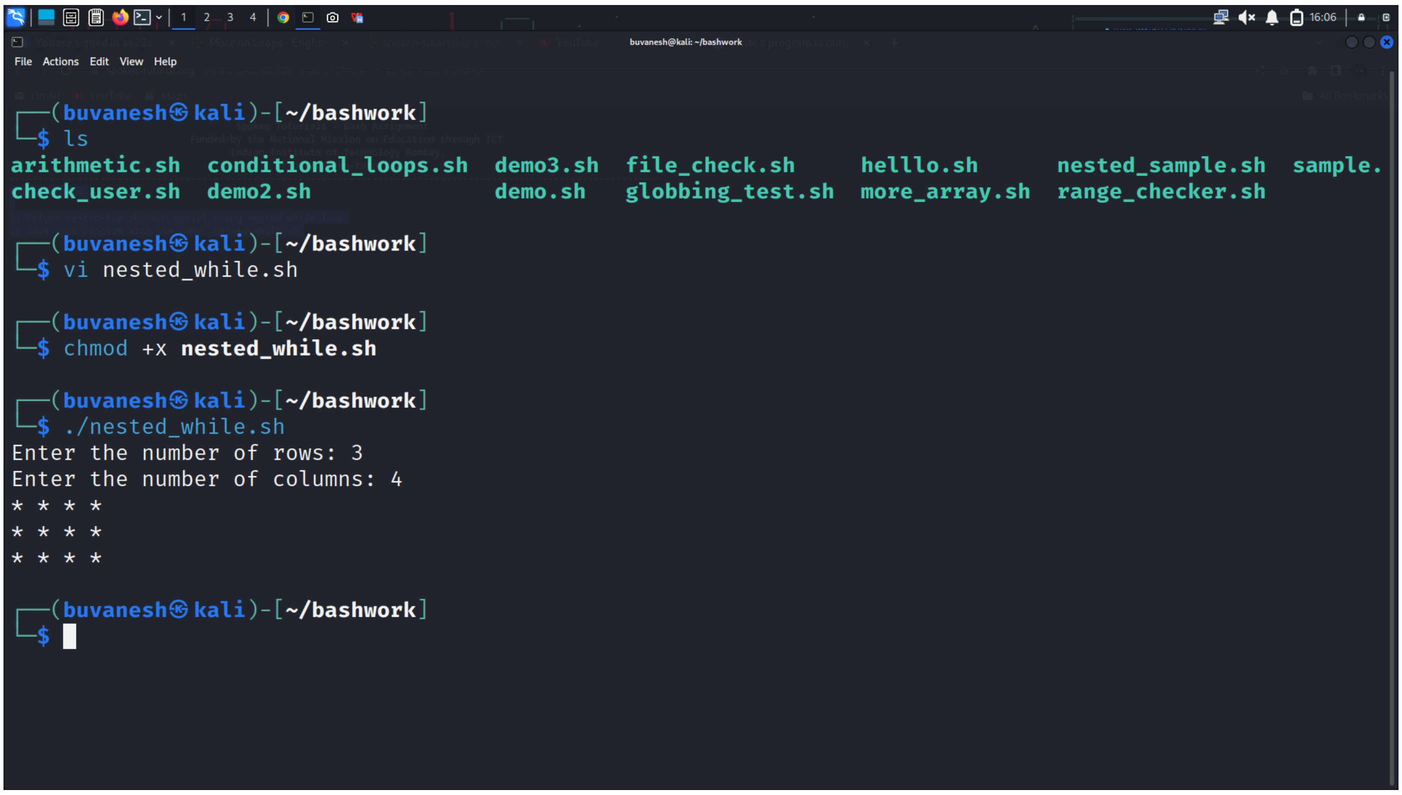
1) Retype nested-for.sh bash script using nested while loop
2) Save your program with the name: nested-while.sh

**script:**



```
#!/bin/bash

# Prompt the user for the number of rows and columns
read -p "Enter the number of rows: " rows
read -p "Enter the number of columns: " columns

# Initialize variables
row=1

# Outer while loop for rows
while ((row <= rows)); do
  column=1

  # Inner while loop for columns
  while ((column <= columns)); do
    echo -n "* "
    column=$((column + 1))
  done

  echo
  row=$((row + 1))
done

~
~
~
~
~
-- INSERT --                                                          23,1          All
```

**output:**

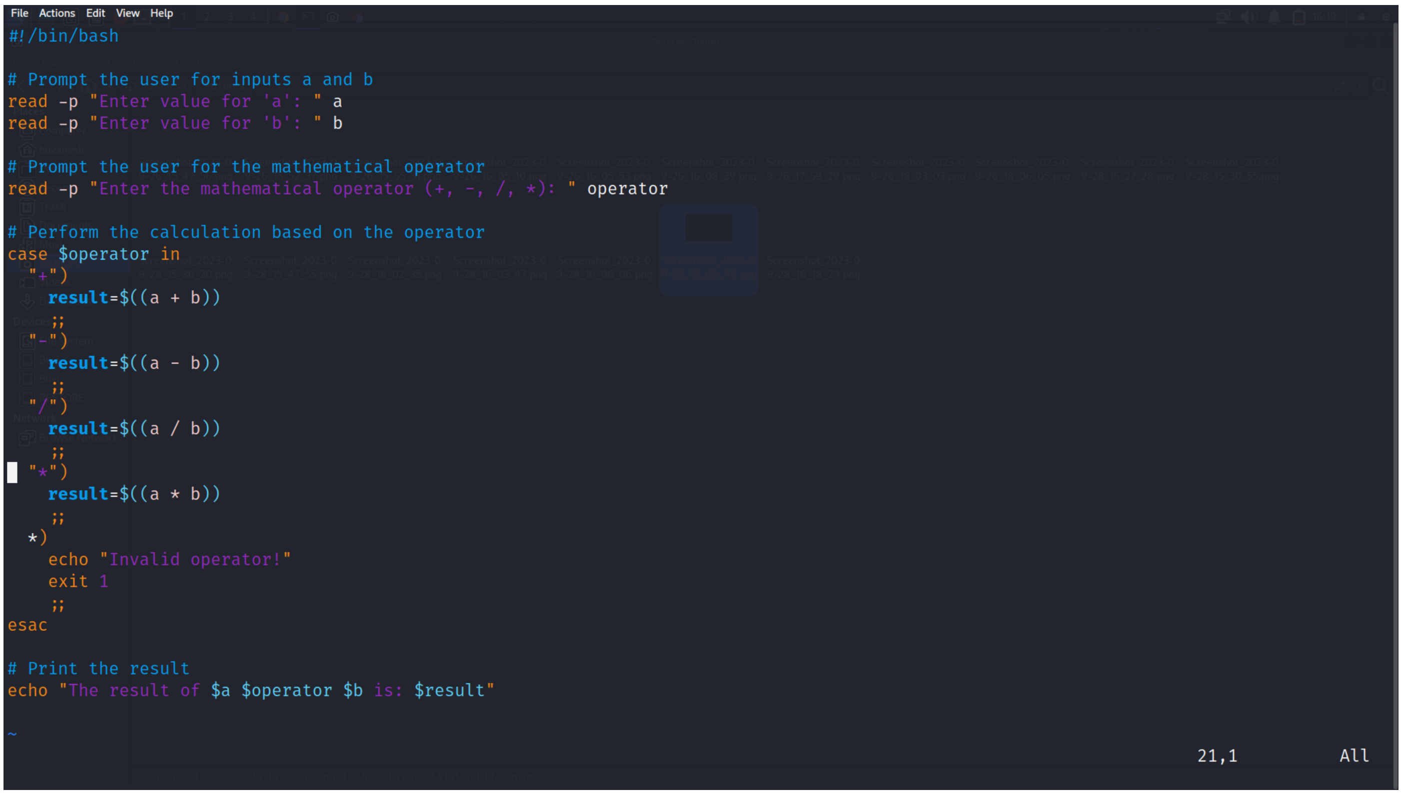


```
┌──(buvanesh㉿kali)-[~/bashwork]
└─$ ls
arithmetic.sh  conditional_loops.sh  demo3.sh  file_check.sh     helllo.sh      nested_sample.sh  sample.
check_user.sh  demo2.sh              demo.sh   globbing_test.sh  more_array.sh  range_checker.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ vi nested_while.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ chmod +x nested_while.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ ./nested_while.sh
Enter the number of rows: 3
Enter the number of columns: 4
* * * *
* * * *
* * * *

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ 
```

## Case statement

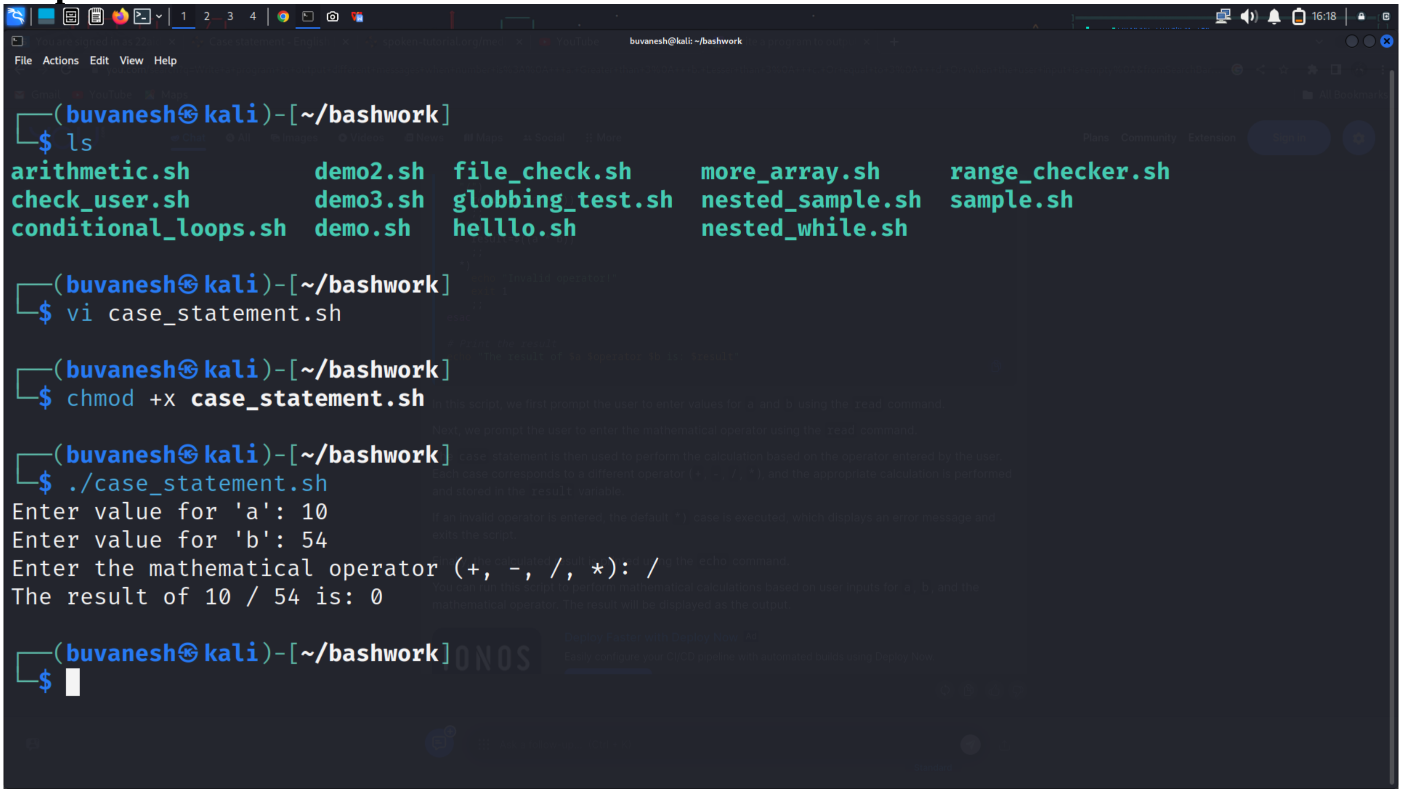
1) Write a menu driven program for mathematical calculation
   a. It should take user inputs a and b
   b. It should ask for mathematical operator (+, -, / and *).
   c. Do the calculation
   d. Print the output

**script:**



```
#!/bin/bash

# Prompt the user for inputs a and b
read -p "Enter value for 'a': " a
read -p "Enter value for 'b': " b

# Prompt the user for the mathematical operator
read -p "Enter the mathematical operator (+, -, /, *): " operator

# Perform the calculation based on the operator
case $operator in
  "+")
    result=$((a + b))
    ;;
  "-")
    result=$((a - b))
    ;;
  "/")
    result=$((a / b))
    ;;
  "*")
    result=$((a * b))
    ;;
  *)
    echo "Invalid operator!"
    exit 1
    ;;
esac

# Print the result
echo "The result of $a $operator $b is: $result"
~
                                                              21,1          All
```

**output:**

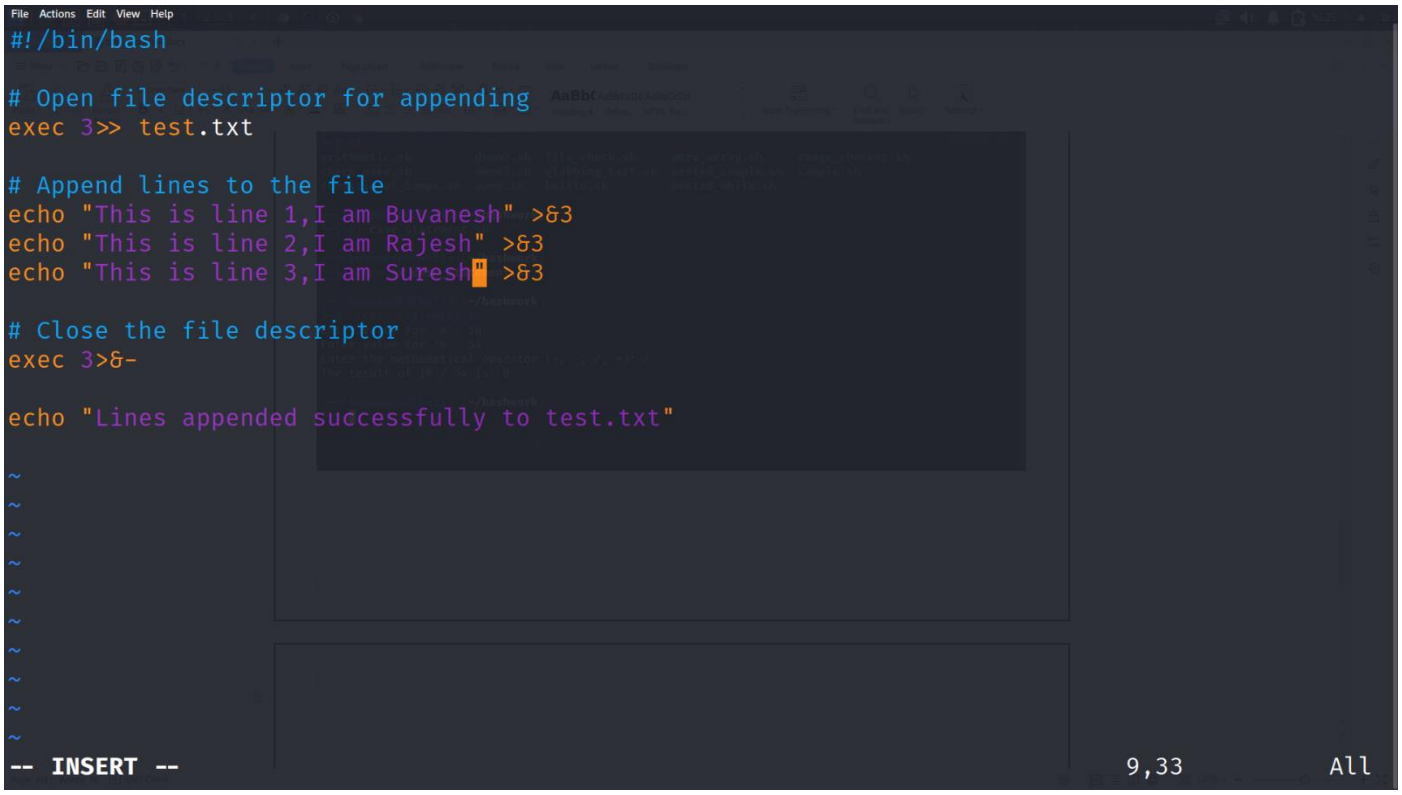


```
┌──(buvanesh㉿kali)-[~/bashwork]
└─$ ls
arithmetic.sh          demo2.sh  file_check.sh     more_array.sh     range_checker.sh
check_user.sh          demo3.sh  globbing_test.sh  nested_sample.sh  sample.sh
conditional_loops.sh   demo.sh   helllo.sh         nested_while.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ vi case_statement.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ chmod +x case_statement.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ ./case_statement.sh
Enter value for 'a': 10
Enter value for 'b': 54
Enter the mathematical operator (+, -, /, *): /
The result of 10 / 54 is: 0

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ 
```

## Using File Descriptors

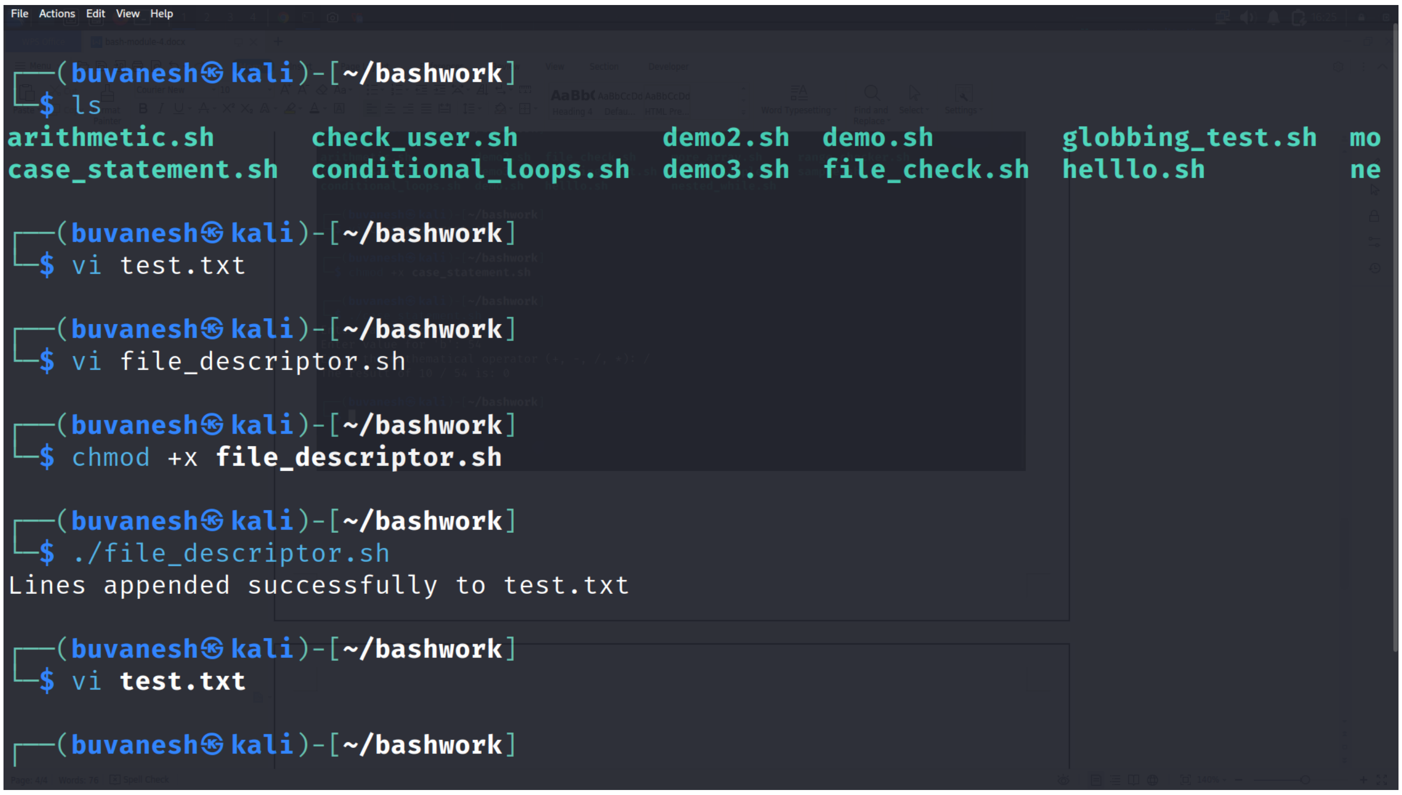
1) Try to append few lines to a file test.txt using file descriptor.
2) Display the content of the file using file descriptor.

**script:**



```
#!/bin/bash

# Open file descriptor for appending
exec 3>> test.txt

# Append lines to the file
echo "This is line 1,I am Buvanesh" >&3
echo "This is line 2,I am Rajesh" >&3
echo "This is line 3,I am Suresh" >&3

# Close the file descriptor
exec 3>&-

echo "Lines appended successfully to test.txt"

~
~
~
~
~
~
~
~
~
~
-- INSERT --                                               9,33          All
```

**output:**

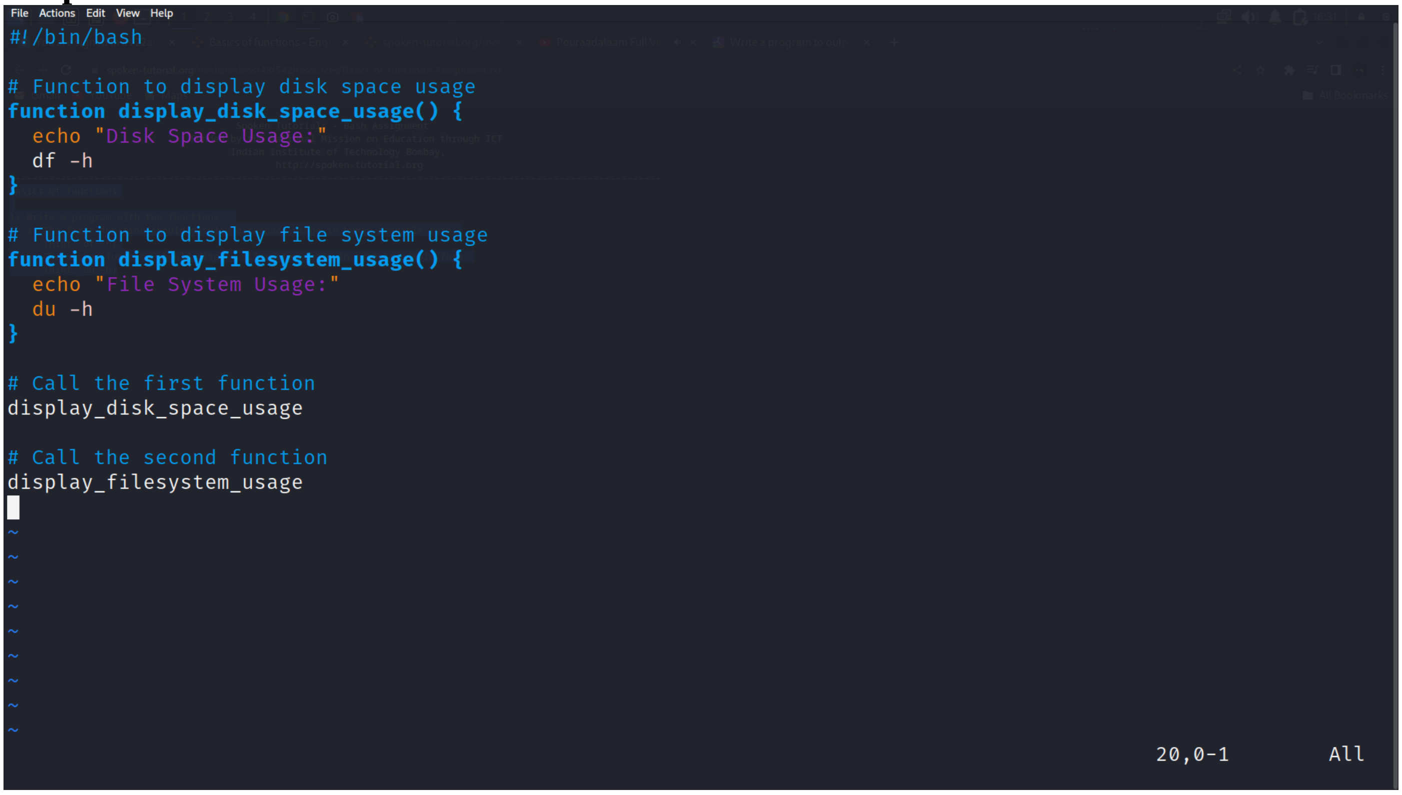


```
┌──(buvanesh㉿kali)-[~/bashwork]
└─$ ls
arithmetic.sh      check_user.sh         demo2.sh  demo.sh        globbing_test.sh  mo
case_statement.sh  conditional_loops.sh  demo3.sh  file_check.sh  helllo.sh         ne

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ vi test.txt

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ vi file_descriptor.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ chmod +x file_descriptor.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ ./file_descriptor.sh
Lines appended successfully to test.txt

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ vi test.txt

┌──(buvanesh㉿kali)-[~/bashwork]
```

## Basics of functions

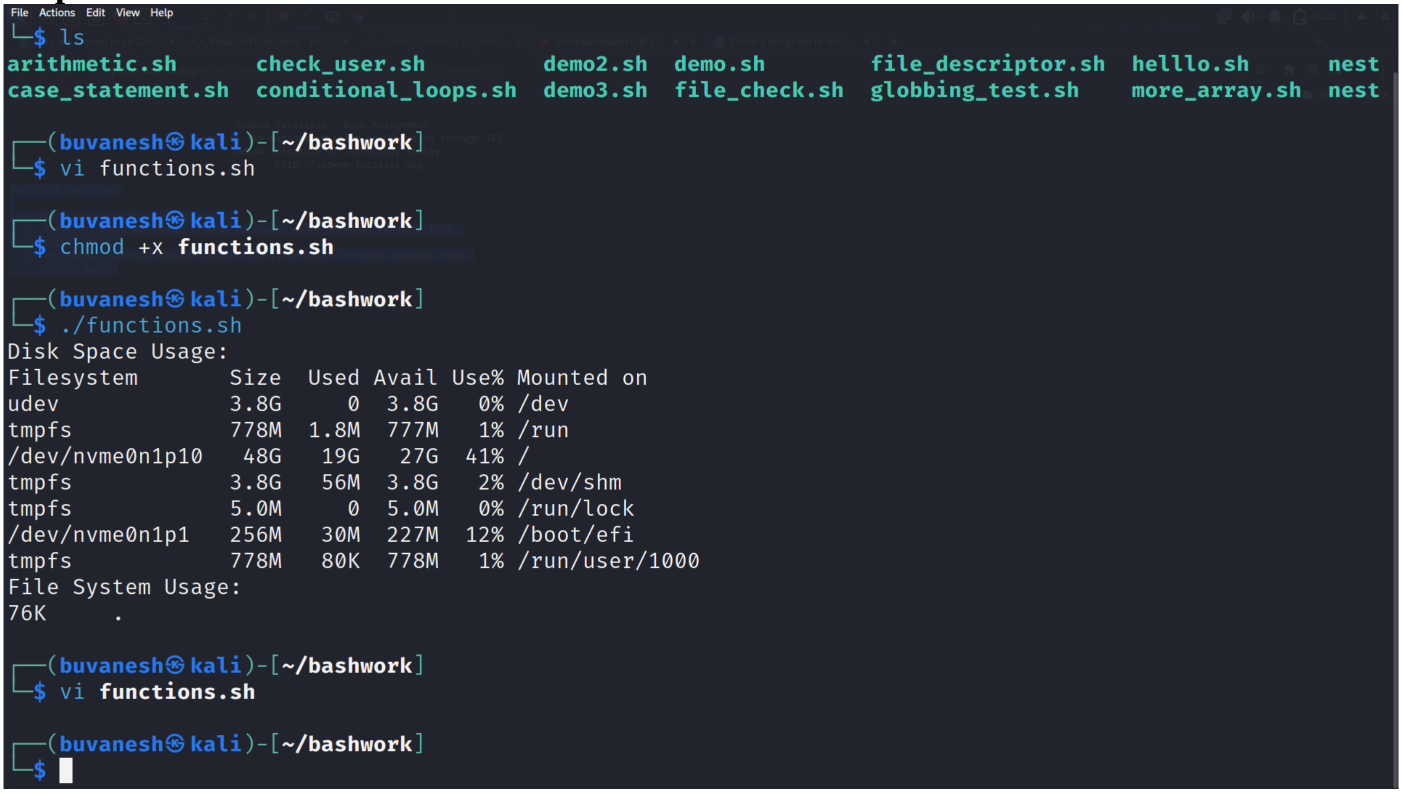
1) Write a program with two functions:
   a. The first function should display diskspace usage in human readable form.
      (Hint: df -h)
   b. The second function should display filesystem usage in human readable form.
      (Hint: du -h)

**script:**



```
#!/bin/bash

# Function to display disk space usage
function display_disk_space_usage() {
  echo "Disk Space Usage:"
  df -h
}

# Function to display file system usage
function display_filesystem_usage() {
  echo "File System Usage:"
  du -h
}

# Call the first function
display_disk_space_usage

# Call the second function
display_filesystem_usage

~
~
~
~
~
~
~
~
~
                                                                                              20,0-1        All
```

**output:**

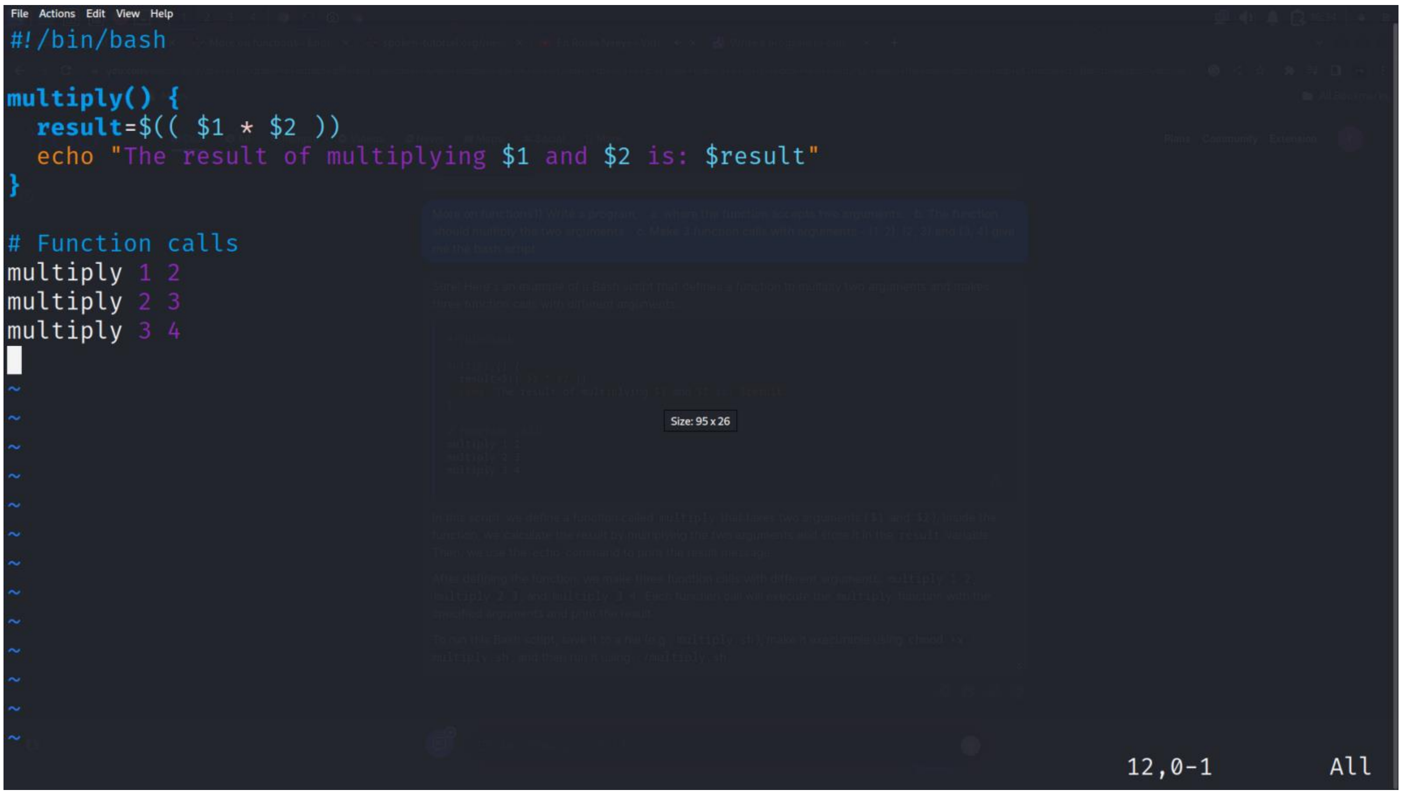


```
└─$ ls
arithmetic.sh      check_user.sh          demo2.sh  demo.sh        file_descriptor.sh  helllo.sh      nest
case_statement.sh  conditional_loops.sh   demo3.sh  file_check.sh  globbing_test.sh    more_array.sh  nest

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ vi functions.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ chmod +x functions.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ ./functions.sh
Disk Space Usage:
Filesystem       Size  Used Avail Use% Mounted on
udev             3.8G     0  3.8G   0% /dev
tmpfs            778M  1.8M  777M   1% /run
/dev/nvme0n1p10   48G   19G   27G  41% /
tmpfs            3.8G   56M  3.8G   2% /dev/shm
tmpfs            5.0M     0  5.0M   0% /run/lock
/dev/nvme0n1p1   256M   30M  227M  12% /boot/efi
tmpfs            778M   80K  778M   1% /run/user/1000
File System Usage:
76K     .

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ vi functions.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ 
```

## More on functions

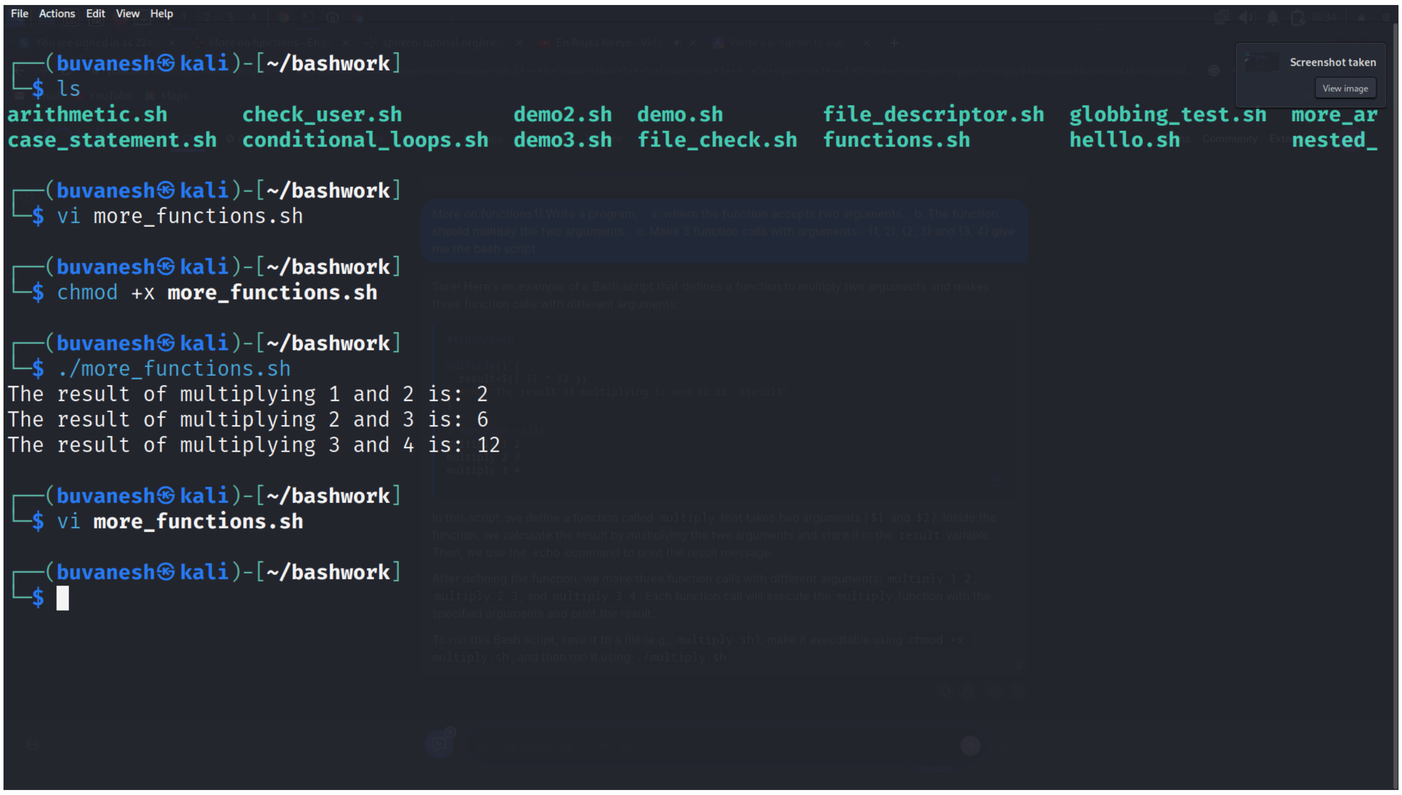
1) Write a program,
   a. where the function accepts two arguments.
   b. The function should multiply the two arguments.
   c. Make 3 function calls with arguments - (1, 2), (2, 3) and (3, 4)

**script:**



```
#!/bin/bash

multiply() {
  result=$(( $1 * $2 ))
  echo "The result of multiplying $1 and $2 is: $result"
}

# Function calls
multiply 1 2
multiply 2 3
multiply 3 4

~
~
~
~
~
~
~
~
~
~
~
~
~
~
                                         12,0-1        All
```

**output:**

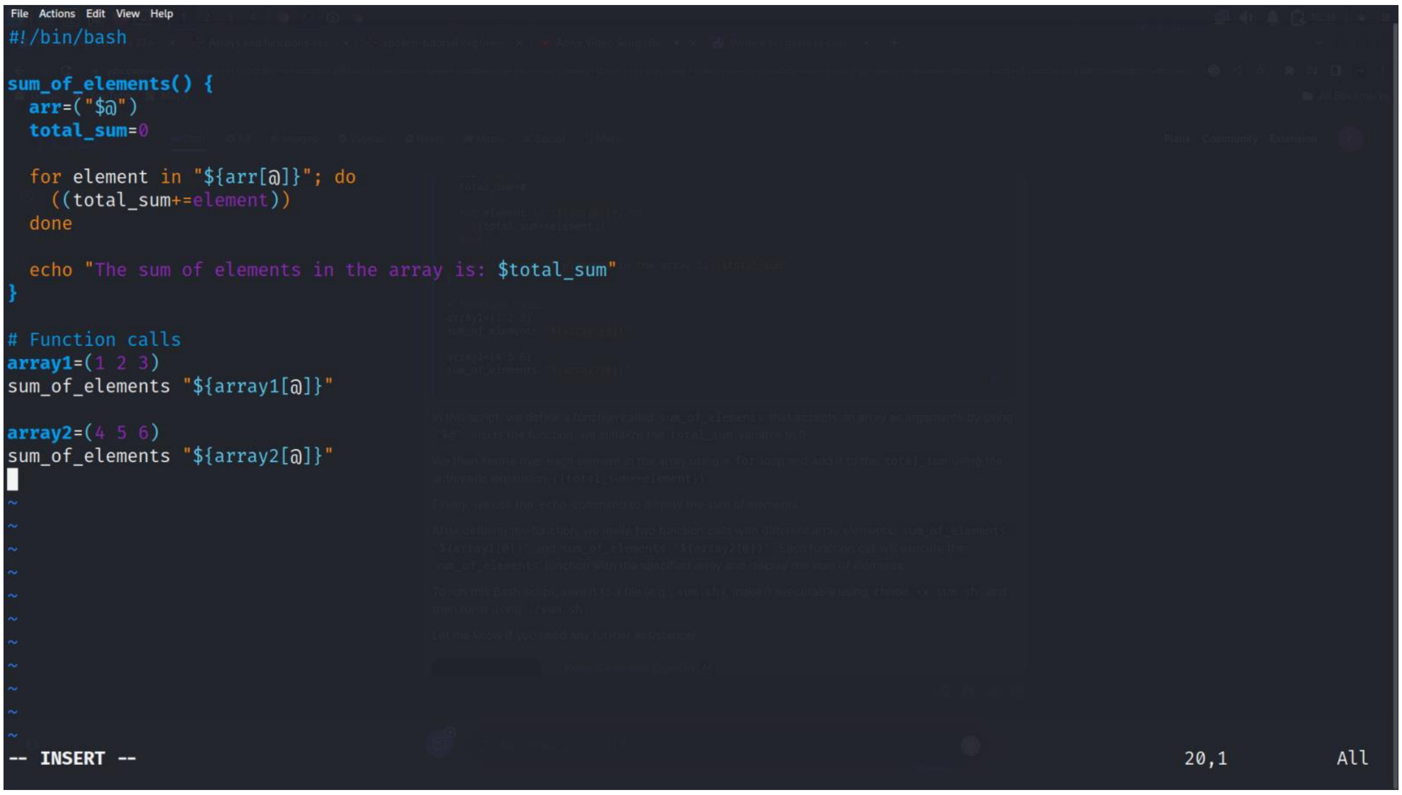


```
┌──(buvanesh㉿kali)-[~/bashwork]
└─$ ls
arithmetic.sh      check_user.sh         demo2.sh  demo.sh        file_descriptor.sh  globbing_test.sh  more_ar
case_statement.sh  conditional_loops.sh  demo3.sh  file_check.sh  functions.sh        helllo.sh         nested_

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ vi more_functions.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ chmod +x more_functions.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ ./more_functions.sh
The result of multiplying 1 and 2 is: 2
The result of multiplying 2 and 3 is: 6
The result of multiplying 3 and 4 is: 12

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ vi more_functions.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ 
```

## Arrays and functions

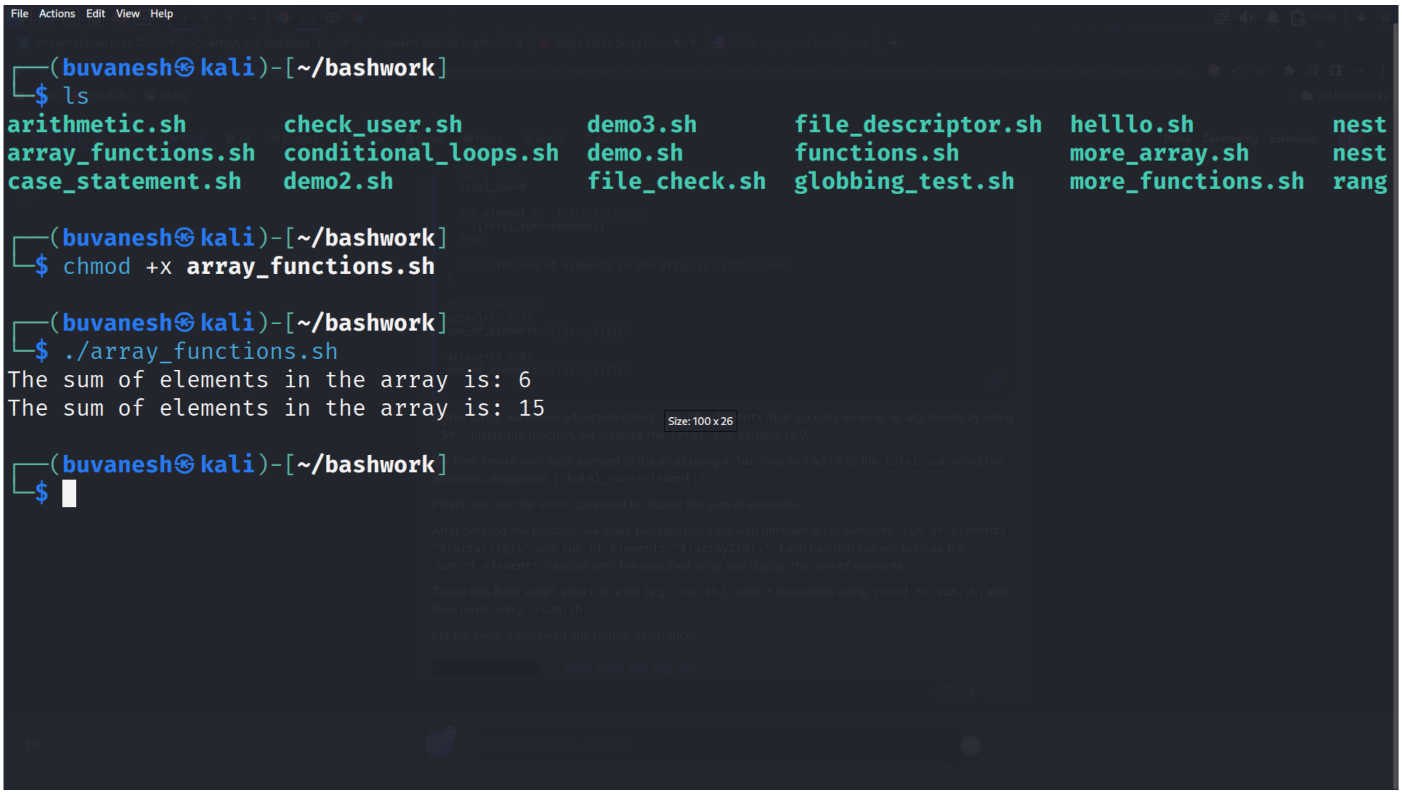
1) Write a program,
   a. Where a function adds all the elements in an array.
   b. The function should display the sum of elements.
   c. Make 2 function calls with array elements- (1, 2, 3) and (4, 5, 6).

**script:**



```
#!/bin/bash

sum_of_elements() {
  arr=("$@")
  total_sum=0

  for element in "${arr[@]}"; do
    ((total_sum+=element))
  done

  echo "The sum of elements in the array is: $total_sum"
}

# Function calls
array1=(1 2 3)
sum_of_elements "${array1[@]}"

array2=(4 5 6)
sum_of_elements "${array2[@]}"

~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --                                                20,1          All
```

**output:**

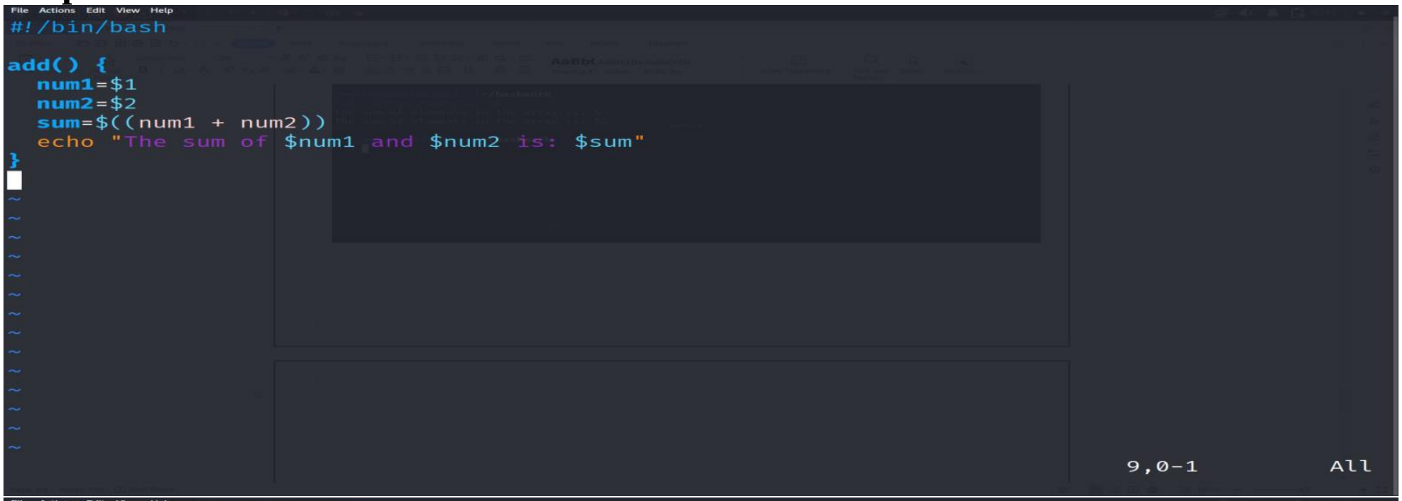


```
┌──(buvanesh㉿kali)-[~/bashwork]
└─$ ls
arithmetic.sh       check_user.sh        demo3.sh       file_descriptor.sh  helllo.sh          nest
array_functions.sh  conditional_loops.sh demo.sh        functions.sh        more_array.sh      nest
case_statement.sh   demo2.sh             file_check.sh  globbing_test.sh    more_functions.sh  rang

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ chmod +x array_functions.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ ./array_functions.sh
The sum of elements in the array is: 6
The sum of elements in the array is: 15

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ 
```

## Advance topics in a function

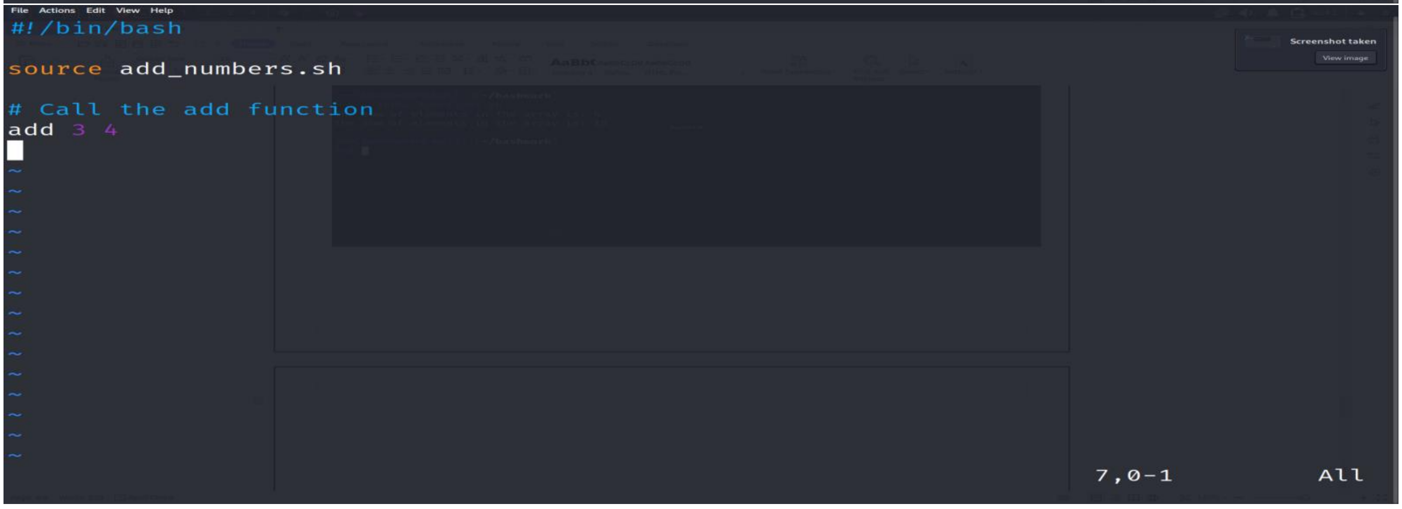
1) Write a function add to add two numbers and call the function in another file.

**script:**



```
#!/bin/bash

add() {
  num1=$1
  num2=$2
  sum=$((num1 + num2))
  echo "The sum of $num1 and $num2 is: $sum"
}
```

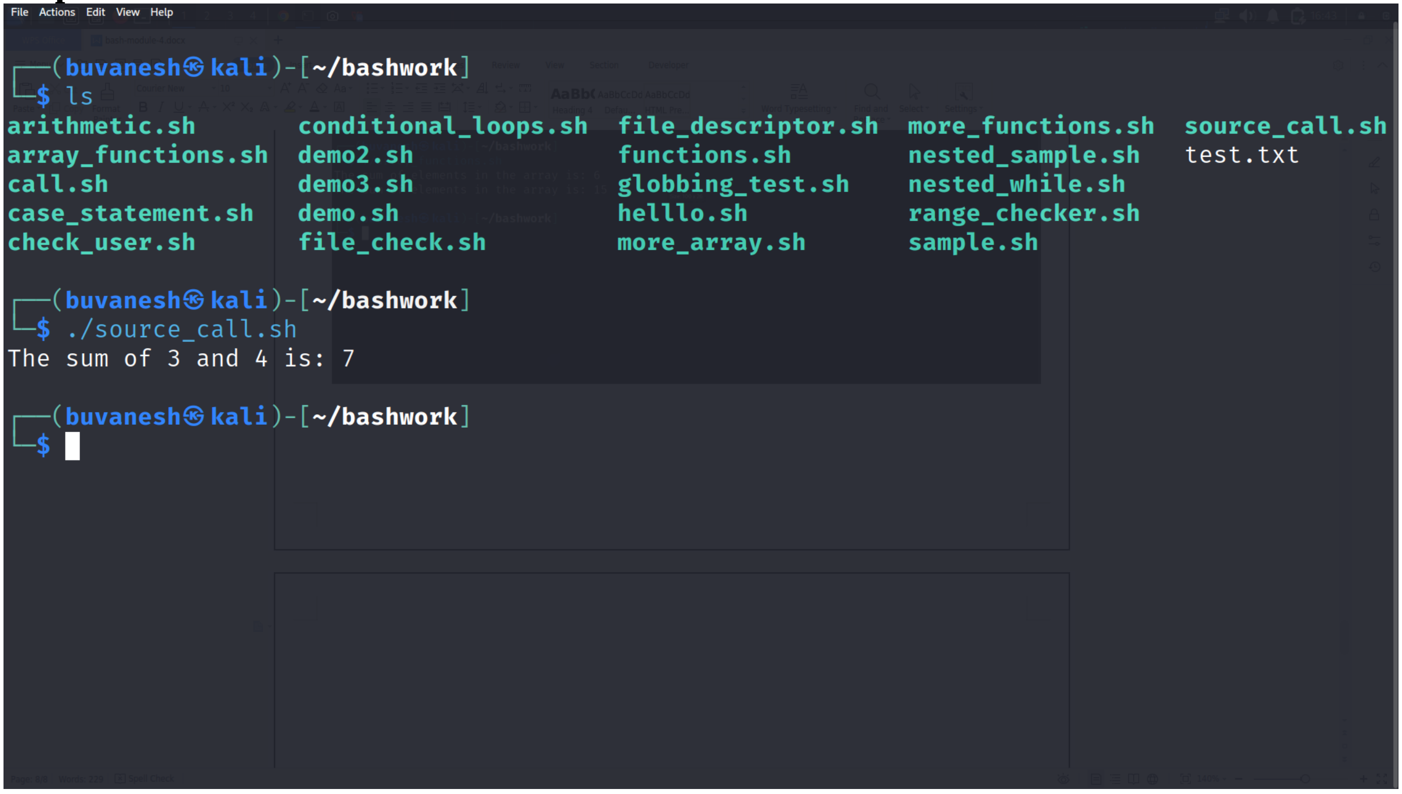
```
9,0-1         All
```



```
#!/bin/bash

source add_numbers.sh

# Call the add function
add 3 4
```

```
7,0-1         All
```

**output:**

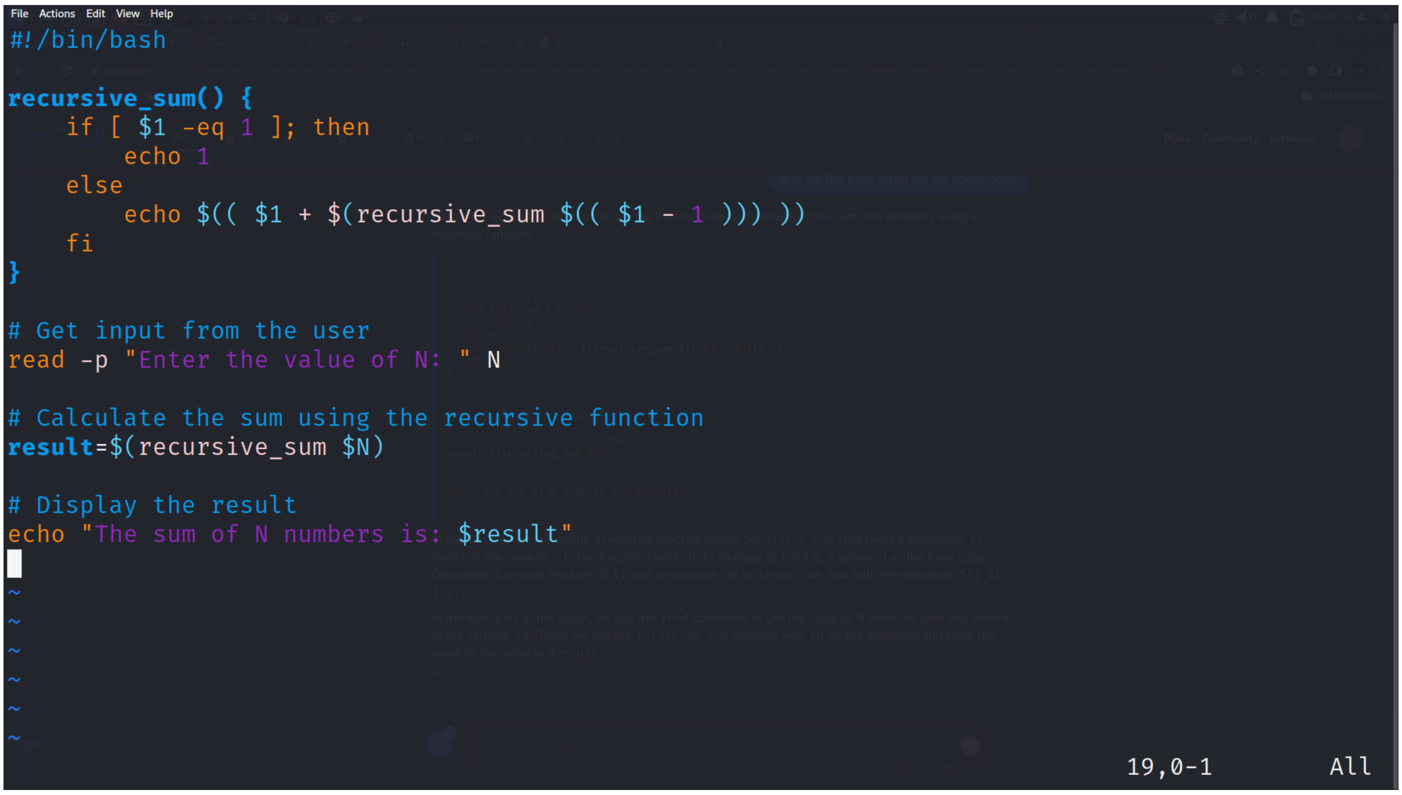


```
┌──(buvanesh㉿kali)-[~/bashwork]
└─$ ls
arithmetic.sh       conditional_loops.sh  file_descriptor.sh  more_functions.sh  source_call.sh
array_functions.sh  demo2.sh              functions.sh        nested_sample.sh   test.txt
call.sh             demo3.sh              globbing_test.sh    nested_while.sh
case_statement.sh   demo.sh               helllo.sh           range_checker.sh
check_user.sh       file_check.sh         more_array.sh       sample.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ ./source_call.sh
The sum of 3 and 4 is: 7

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ 
```

## Recursive function

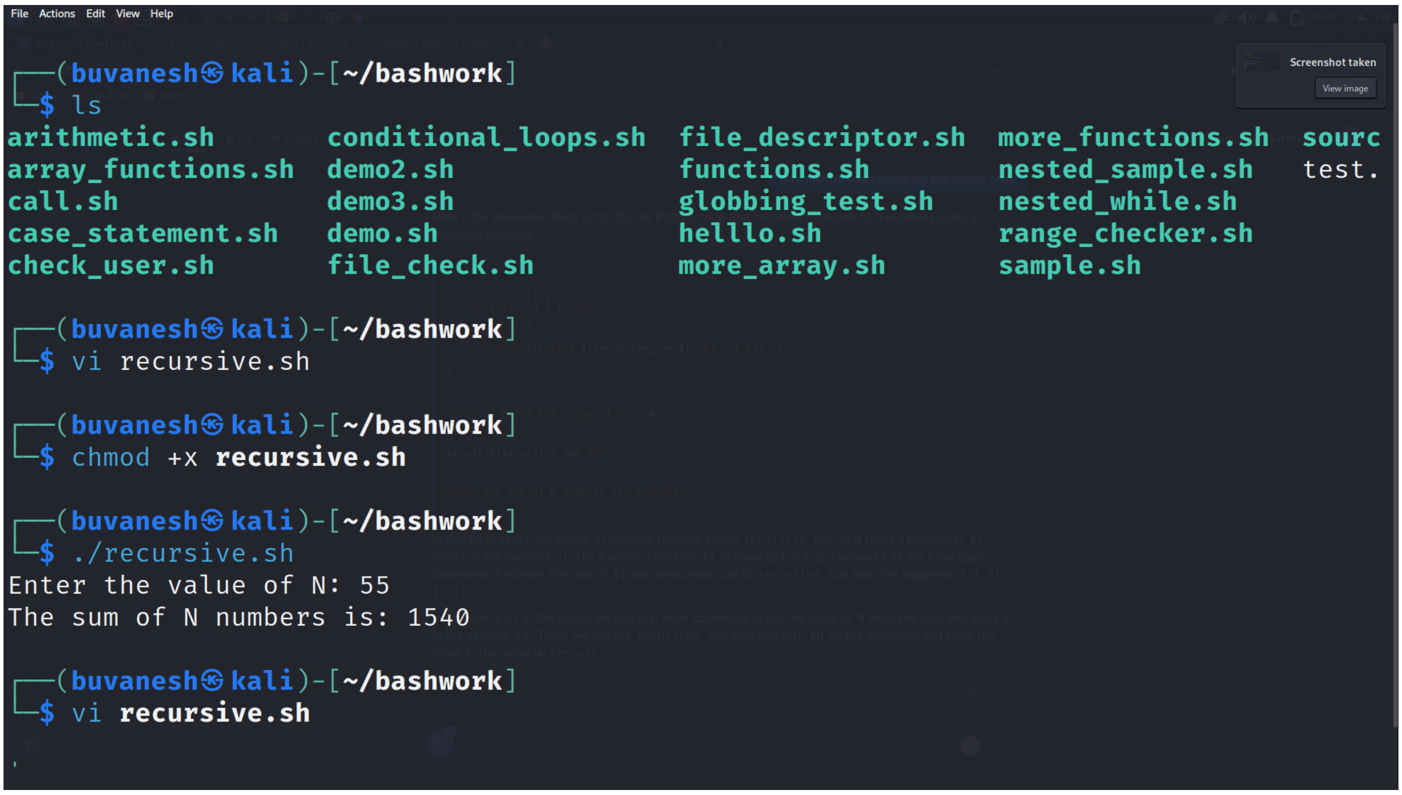
1) Write a program where the recursive function calculates the sum of N numbers

**script:**



```
#!/bin/bash

recursive_sum() {
    if [ $1 -eq 1 ]; then
        echo 1
    else
        echo $(( $1 + $(recursive_sum $(( $1 - 1 ))) ))
    fi
}

# Get input from the user
read -p "Enter the value of N: " N

# Calculate the sum using the recursive function
result=$(recursive_sum $N)

# Display the result
echo "The sum of N numbers is: $result"
~
~
~
~
~
~
                                                            19,0-1        All
```

**output:**

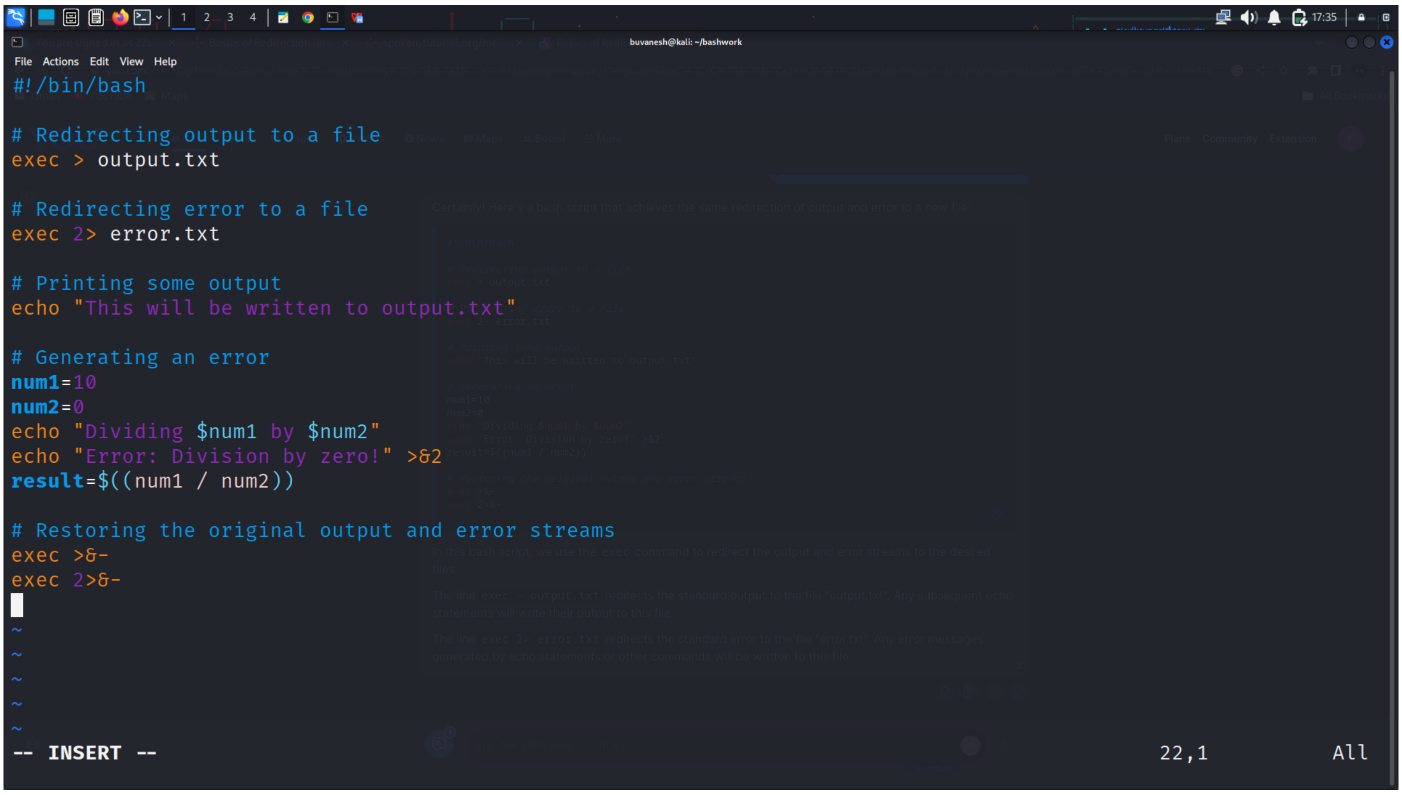


```
┌──(buvanesh㉿kali)-[~/bashwork]
└─$ ls
arithmetic.sh       conditional_loops.sh  file_descriptor.sh  more_functions.sh  sourc
array_functions.sh  demo2.sh              functions.sh        nested_sample.sh   test.
call.sh             demo3.sh              globbing_test.sh    nested_while.sh
case_statement.sh   demo.sh               helllo.sh           range_checker.sh
check_user.sh       file_check.sh         more_array.sh       sample.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ vi recursive.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ chmod +x recursive.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ ./recursive.sh
Enter the value of N: 55
The sum of N numbers is: 1540

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ vi recursive.sh
```

# Basics of Redirection (error handling)

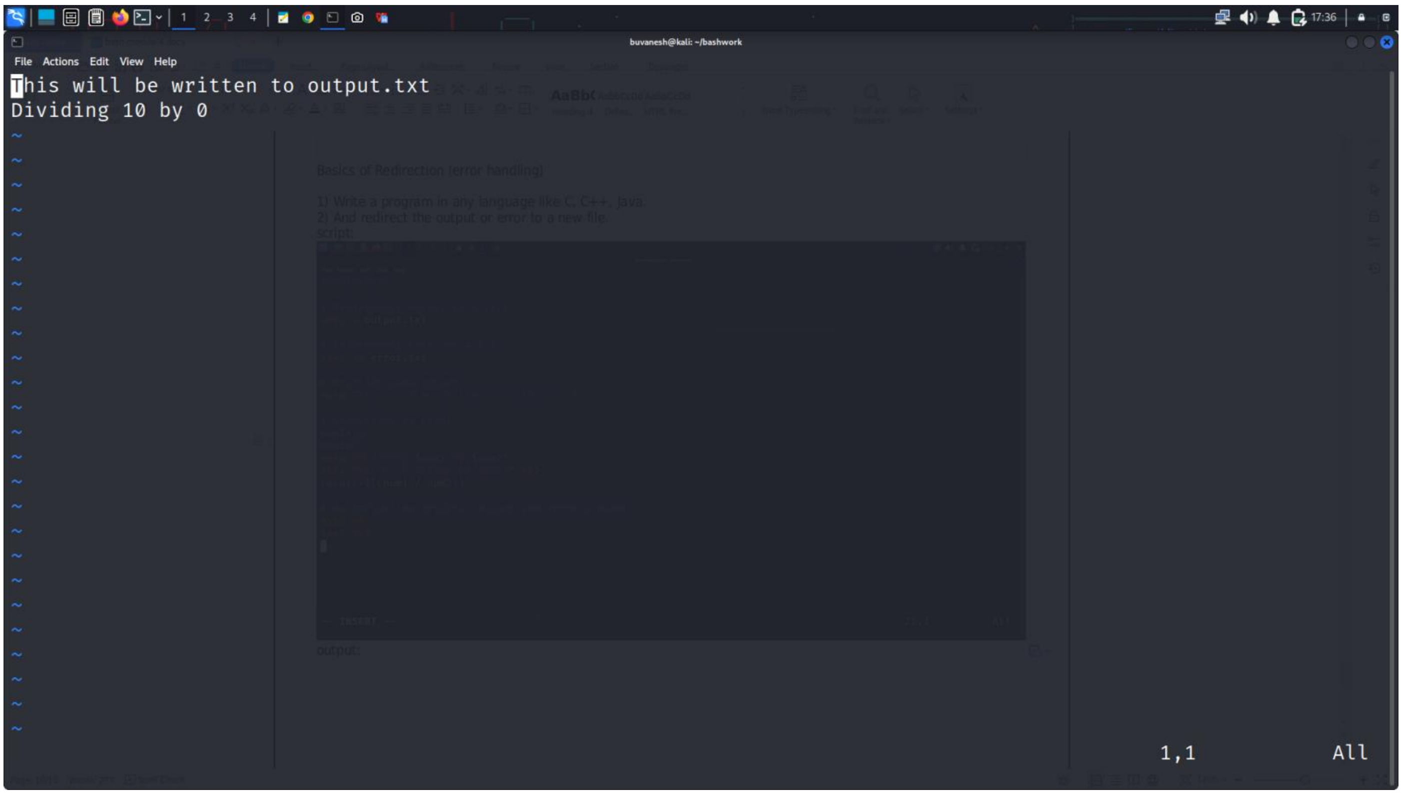
1) Write a program in any language like C, C++, Java.
2) And redirect the output or error to a new file.

**script:**

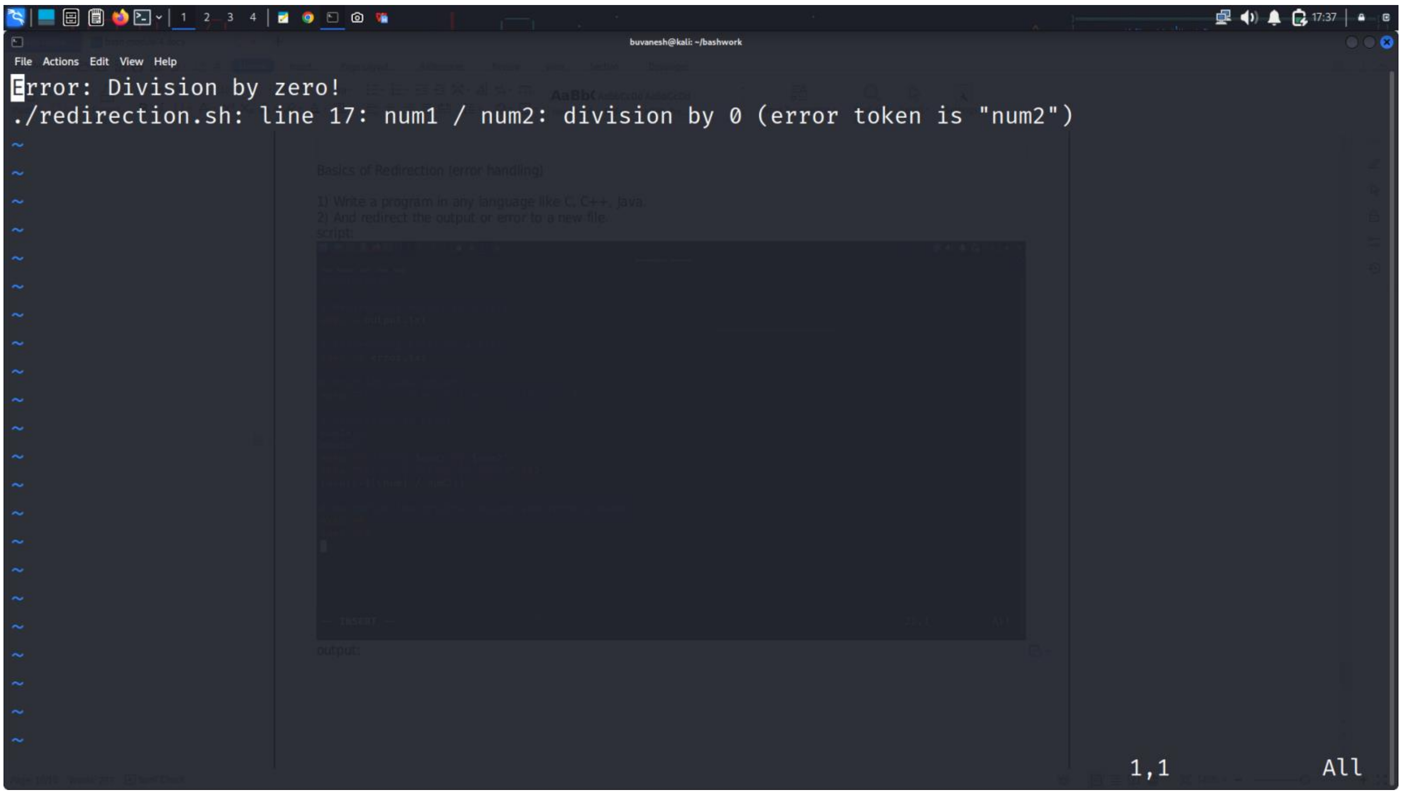


```
#!/bin/bash

# Redirecting output to a file
exec > output.txt

# Redirecting error to a file
exec 2> error.txt

# Printing some output
echo "This will be written to output.txt"

# Generating an error
num1=10
num2=0
echo "Dividing $num1 by $num2"
echo "Error: Division by zero!" >&2
result=$((num1 / num2))

# Restoring the original output and error streams
exec >&-
exec 2>&-

~
~
~
~
~
-- INSERT --                                                 22,1          All
```

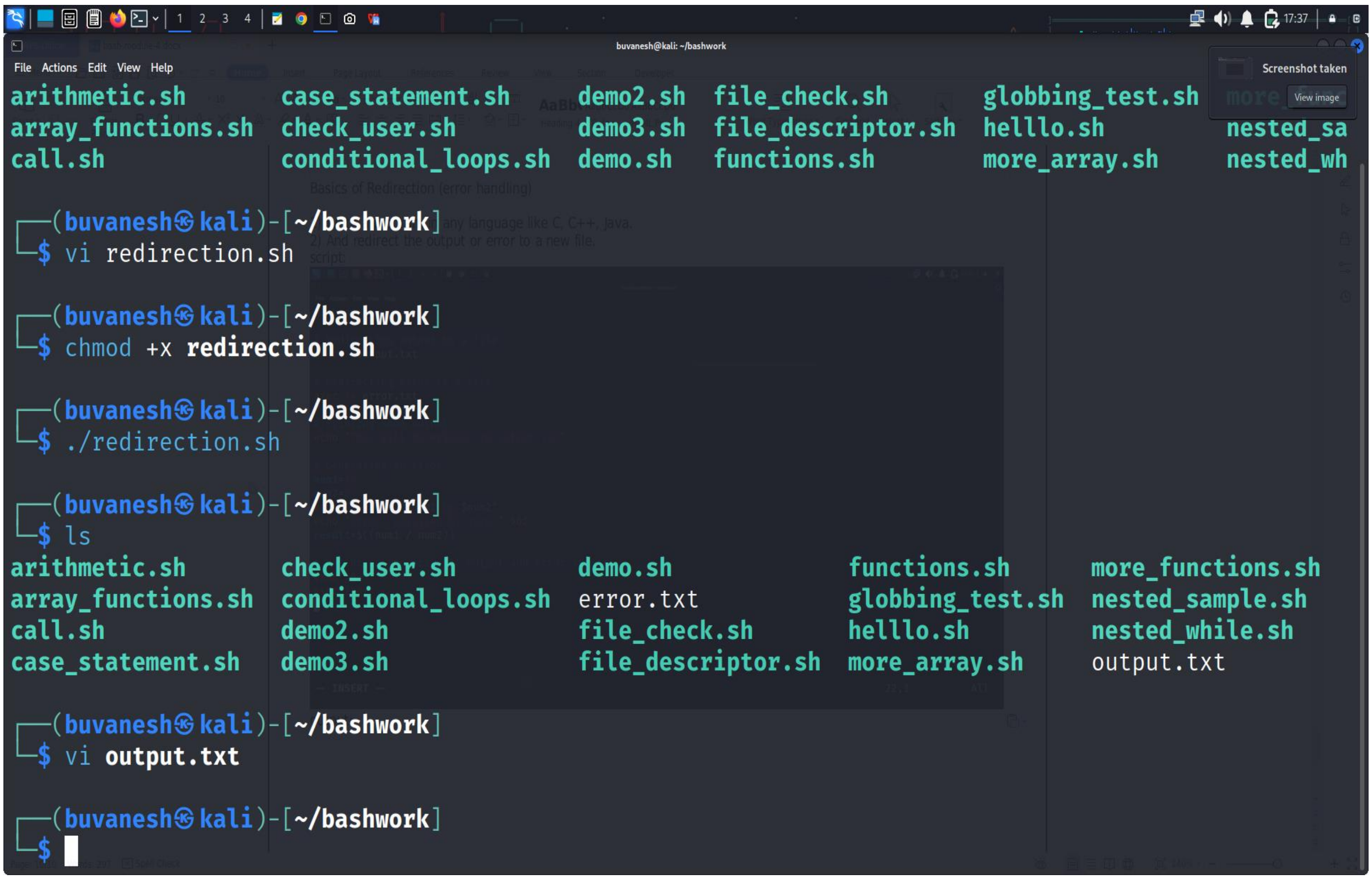
**output:**



```
This will be written to output.txt
Dividing 10 by 0
~
~
~
                                                              1,1           All
```



```
Error: Division by zero!
./redirection.sh: line 17: num1 / num2: division by 0 (error token is "num2")
~
~
~
                                                                          1,1           All
```

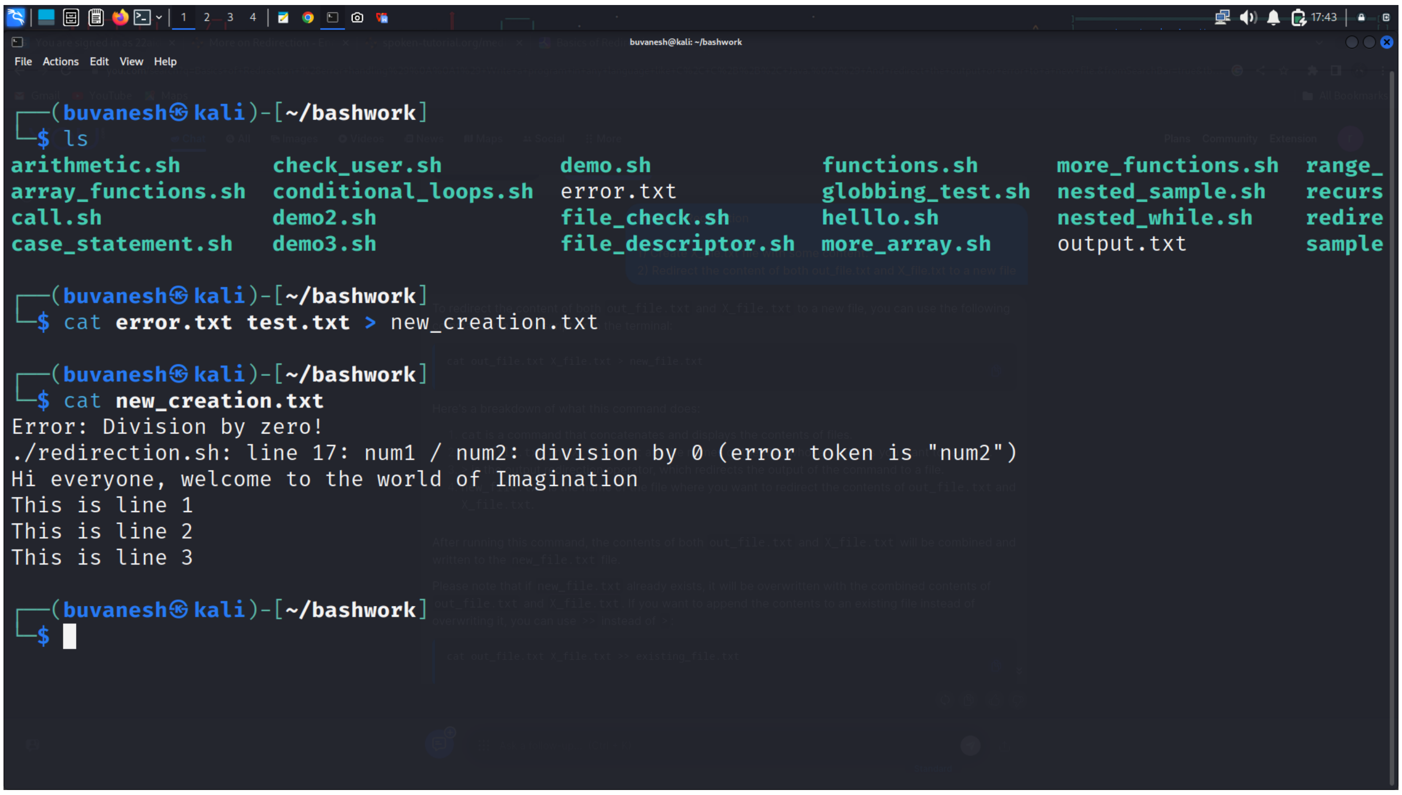


```
arithmetic.sh        case_statement.sh     demo2.sh   file_check.sh        globbing_test.sh   more_
array_functions.sh   check_user.sh         demo3.sh   file_descriptor.sh   helllo.sh          nested_sa
call.sh              conditional_loops.sh  demo.sh    functions.sh         more_array.sh      nested_wh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ vi redirection.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ chmod +x redirection.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ ./redirection.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ ls
arithmetic.sh        check_user.sh         demo.sh              functions.sh       more_functions.sh
array_functions.sh   conditional_loops.sh  error.txt            globbing_test.sh   nested_sample.sh
call.sh              demo2.sh              file_check.sh        helllo.sh          nested_while.sh
case_statement.sh    demo3.sh              file_descriptor.sh   more_array.sh      output.txt

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ vi output.txt

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ 
```

## More on Redirection

1) Create X_file.txt file with some content.
2) Redirect the content of both out_file.txt and X_file.txt to a new file

**output:**

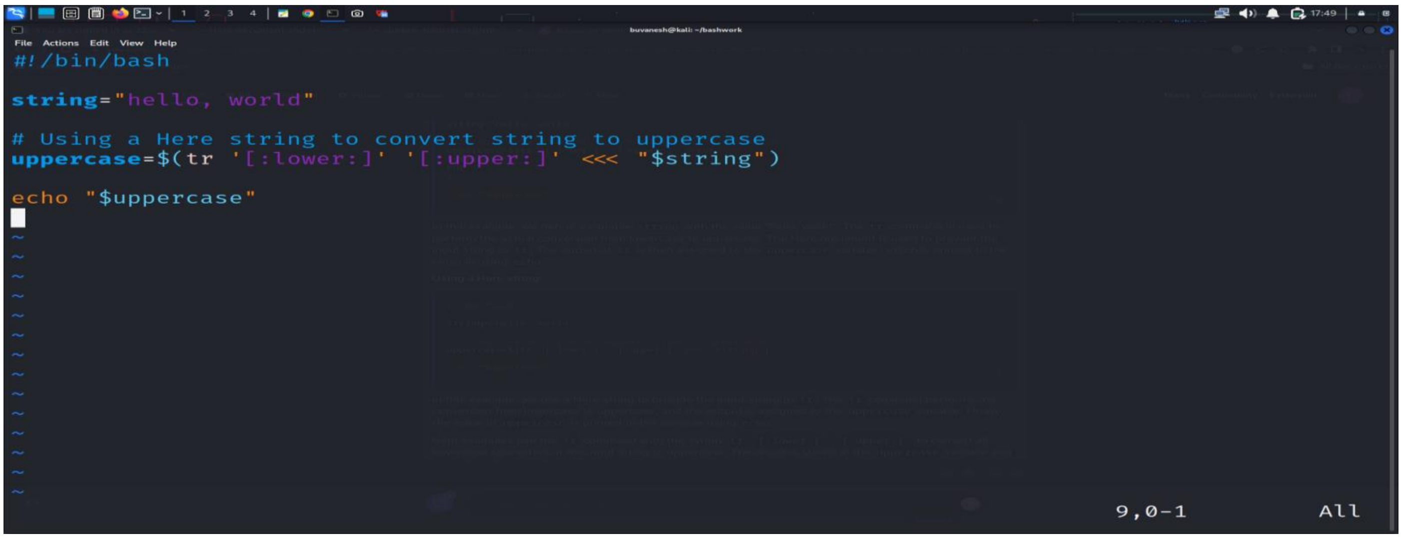


```
┌──(buvanesh㉿kali)-[~/bashwork]
└─$ ls
arithmetic.sh        check_user.sh         demo.sh              functions.sh        more_functions.sh   range_
array_functions.sh   conditional_loops.sh  error.txt            globbing_test.sh    nested_sample.sh    recurs
call.sh              demo2.sh              file_check.sh        helllo.sh           nested_while.sh     redire
case_statement.sh    demo3.sh              file_descriptor.sh   more_array.sh       output.txt          sample

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ cat error.txt test.txt > new_creation.txt

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ cat new_creation.txt
Error: Division by zero!
./redirection.sh: line 17: num1 / num2: division by 0 (error token is "num2")
Hi everyone, welcome to the world of Imagination
This is line 1
This is line 2
This is line 3

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ █
```

## Here document and Here string

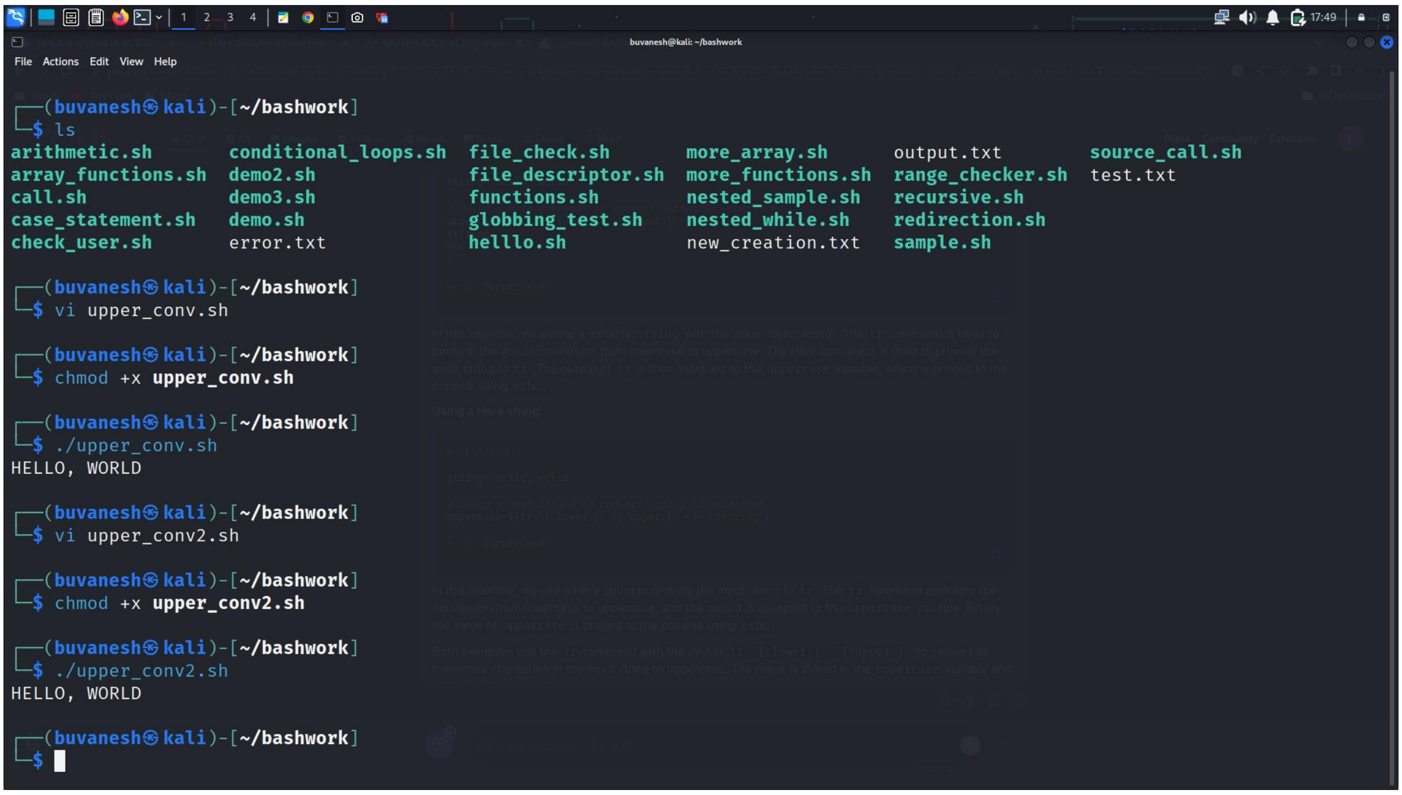
1) Convert a string to uppercase using:
   a) Here document
   b) Here string
   Hint: tr a-z A-Z

**script:**



```
#!/bin/bash

string="hello, world"

# Using a Here string to convert string to uppercase
uppercase=$(tr '[:lower:]' '[:upper:]' <<< "$string")

echo "$uppercase"

~
~
~
~
~
~
~
~
~
~
~
~
~
~
                                                              9,0-1         All
```

**output:**



```
┌──(buvanesh㉿kali)-[~/bashwork]
└─$ ls
arithmetic.sh       conditional_loops.sh  file_check.sh       more_array.sh       output.txt        source_call.sh
array_functions.sh  demo2.sh              file_descriptor.sh  more_functions.sh   range_checker.sh  test.txt
call.sh             demo3.sh              functions.sh        nested_sample.sh    recursive.sh
case_statement.sh   demo.sh               globbing_test.sh    nested_while.sh     redirection.sh
check_user.sh       error.txt             helllo.sh           new_creation.txt    sample.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ vi upper_conv.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ chmod +x upper_conv.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ ./upper_conv.sh
HELLO, WORLD

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ vi upper_conv2.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ chmod +x upper_conv2.sh

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ ./upper_conv2.sh
HELLO, WORLD

┌──(buvanesh㉿kali)-[~/bashwork]
└─$ 
```