**1. Write a Shell program to check the given number is even or odd.**

**Source Code:**

echo "Enter a number: "

read number

if [ $((number % 2)) ]; then

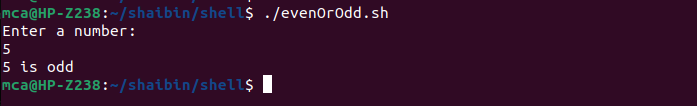
echo "$number is even"

else

echo "$number is odd"

fi

**Output:**



**2. Write a Shell program to check a leap year.**

**Source Code:**

echo "Enter a year: "

read year

if [ $((year % 4)) -eq 0 ] && [ $((year % 100)) -ne 0 ] || [ $((year % 400)) -eq 0 ]; then

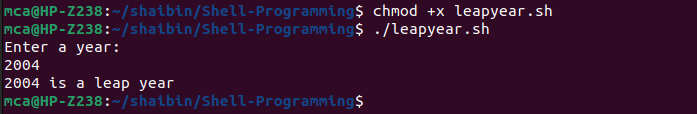
echo "$year is a leap year"

else

echo "$year is not a leap year"

fi

**Output:**



**3.Write a Shell program to find the area and circumference of a circle.**

**Source Code:**

echo "Enter the radious of the circle"

read r

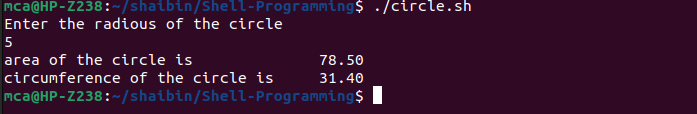
area=$(echo "3.14\*$r\*$r" | bc )

circum=$(echo "3.14\*2\*$r" | bc)

echo "area of the circle is " $area

echo "circumference of the circle is " $circum

**Output:**



**4. Write a Shell program to check the given number and its reverse are same.**

**Source Code:**

echo "Enter a number: "

read number

reverse=0

original=$number

while [ $number -ne 0 ]

do

remainder=$(( $number % 10 ))

reverse=$(( $reverse \* 10 + $remainder ))

number=$(( $number / 10 ))

done

if [ $original -eq $reverse ]

then

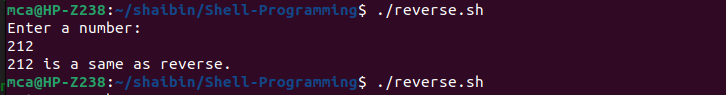
echo "$original is a same as reverse."

else

echo "$original is not same as reverse."

fi

**Output:**



**5. Write a Shell program to check the given string is palindrome or not.**

**Source Code:**

read -p "Enter a string: " input\_string

if [ "$input\_string" == "$(rev <<< "$input\_string")" ]; then

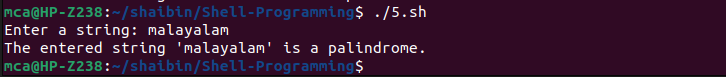
echo "The entered string '$input\_string' is a palindrome."

else

echo "The entered string '$input\_string' is not a palindrome."

fi

**Output:**



**6. Write a Shell program to find the sum of odd and even numbers from a set of numbers.**

**Source Code:**

echo "Enter the size of the array:"

read size

echo "Enter $size values into the array:"

for ((i = 0; i < size; i++)); do

read array[$i]

done

even=0

odd=0

for ((i = 0; i < size; i++)); do

if ((${array[$i]} % 2 == 0)); then

even=$((even + ${array[$i]}))

else

odd=$((odd + ${array[$i]}))

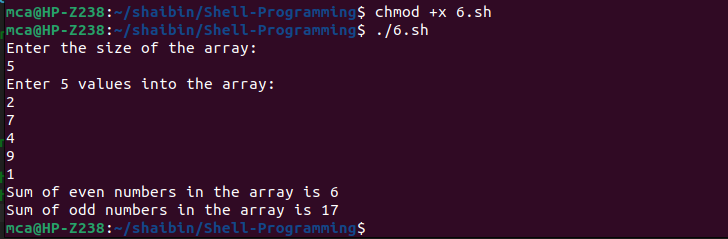
fi

done

echo "Sum of even numbers in the array is $even"

echo "Sum of odd numbers in the array is $odd"

**Output:**



**7.Write a Shell program to find the roots of a quadratic equation.**

**Source Code:**

echo "Enter the coefficients (a, b, c) of the quadratic equation (ax^2 + bx + c):"

read a b c

discriminant=$((b \* b - 4 \* a \* c))

if [ $discriminant -gt 0 ]; then

root1=$(echo "(-$b + sqrt($discriminant)) / (2 \* $a)" | bc -l)

root2=$(echo "(-$b - sqrt($discriminant)) / (2 \* $a)" | bc -l)

echo "The roots are real and different."

echo "Root 1 = $root1, Root 2 = $root2"

elif [ $discriminant -eq 0 ]; then

root=$(echo "-$b / (2 \* $a)" | bc -l)

echo "The roots are real and equal."

echo "Root 1 = Root 2 = $root"

else

real\_part=$(echo "-$b / (2 \* $a)" | bc -l)

imaginary\_part=$(echo "sqrt($((-1 \* discriminant))) / (2 \* $a)" | bc -l)

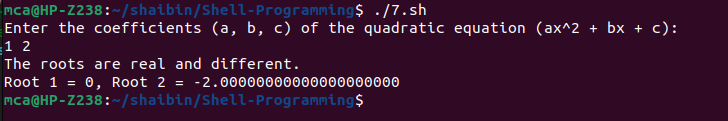
echo "The roots are complex and different."

echo "Root 1 = $real\_part + $imaginary\_part i"

echo "Root 2 = $real\_part - $imaginary\_part i"

fi

**Output:**



**8.Write a Shell program to check the given integer is Armstrong number or not.**

**Source Code:**

echo "Enter an integer:"

read number

num\_digits=${#number}

sum=0

temp=$number

while [ $temp -gt 0 ]; do

digit=$((temp % 10))

sum=$((sum + digit \*\* num\_digits))

temp=$((temp / 10))

done

if [ $sum -eq $number ]; then

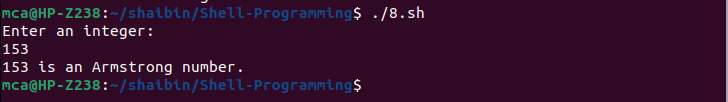
echo "$number is an Armstrong number."

else

echo "$number is not an Armstrong number."

fi

**Output:**



**9.Write a Shell program to check the given integer is prime or not.**

**Source Code:**

echo "Enter an integer:"

read number

factors=0

for ((i = 1; i <= number; i++)); do

if [ $((number % i)) -eq 0 ]; then

factors=$((factors + 1))

fi

done

if [ $factors -eq 2 ]; then

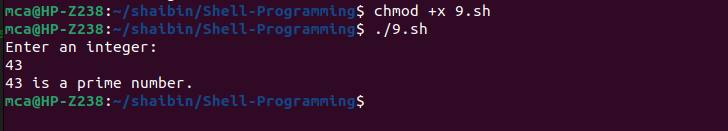
echo "$number is a prime number."

else

echo "$number is not a prime number."

fi

**Output:**



**10.Write a Shell program to generate prime numbers between 1 and 50.**

**Source Code:**

echo "Prime numbers between 1 and 50 are:"

for ((num = 2; num <= 50; num++)); do

is\_prime=true

for ((divisor = 2; divisor < num; divisor++)); do

if [ $((num % divisor)) -eq 0 ]; then

is\_prime=false

break

fi

done

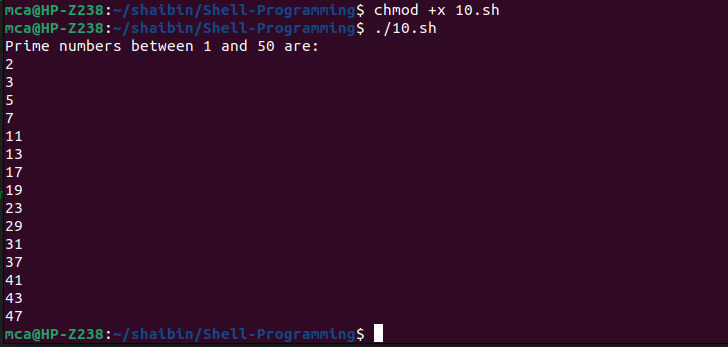
if $is\_prime; then

echo "$num"

fi

done

**Output:**



**11.Write a Shell program to find the sum of square of individual digits of a number.**

**Source Code:**

echo "Enter a number:"

read number

sum=0

while [ $number -gt 0 ]; do

digit=$((number % 10))

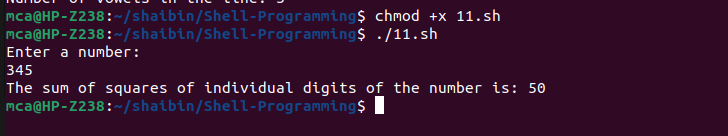
sum=$((sum + digit \* digit))

number=$((number / 10))

done

echo "The sum of squares of individual digits of the number is: $sum"

**Output:**



**12.Write a Shell program to count the number of vowels in a line of text.**

**Source Code:**

echo "Enter a line of text:"

read line

line=$(echo "$line" | tr '[:upper:]' '[:lower:]')

count=0

for ((i = 0; i < ${#line}; i++)); do

char="${line:$i:1}"

case $char in

[aeiou])

count=$((count + 1))

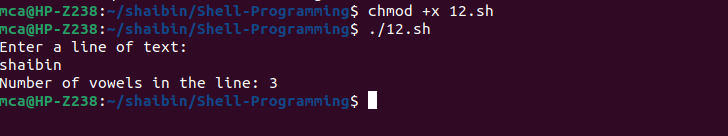
;;

esac

done

echo "Number of vowels in the line: $count"

**Output:**



**12. Write a Shell program to display student grades.**

**Source Code:**

echo "Enter the student's score: "

read score

if ! [[ $score =~ ^[0-9]+$ ]]; then

echo "Invalid input. Please enter a valid number."

exit 1

fi

if (( score >= 90 )); then

grade="A"

elif (( score >= 80 )); then

grade="B"

elif (( score >= 70 )); then

grade="C"

elif (( score >= 60 )); then

grade="D"

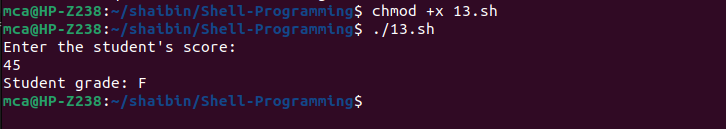
else

grade="F"

fi

echo "Student grade: $grade"

**Output:**



**13. Write a Shell program to find the smallest and largest numbers from a set of numbers.**

**Source Code:**

echo "Enter a list of numbers separated by spaces:"

read -a numbers

sorted\_numbers=($(printf "%s\n" "${numbers[@]}" | sort -n))

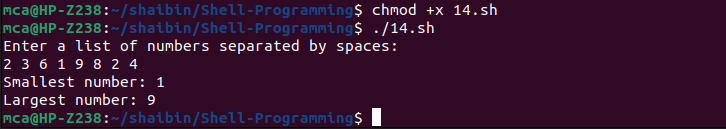
smallest=${sorted\_numbers[0]}

largest=${sorted\_numbers[-1]}

echo "Smallest number: $smallest"

echo "Largest number: $largest"

**Output:**



**14. Write a Shell program to find the smallest digit from a number.**

**Source Code:**

echo "Enter a number:"

read number

smallest\_digit=${number:0:1}

for (( i = 1; i < ${#number}; i++ )); do

digit=${number:i:1}

if [ "$digit" -lt "$smallest\_digit" ]; then

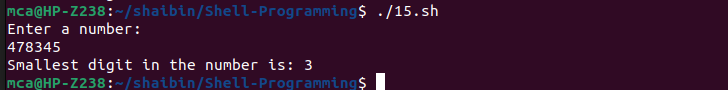
smallest\_digit=$digit

fi

done

echo "Smallest digit in the number is: $smallest\_digit"

**Output:**



**15. Write a Shell program to find the sum of all numbers between 50 and 100, which are divisible by 3 and not divisible by 5.**

**Source Code:**

sum=0

for (( num = 50; num <= 100; num++ )); do

if (( num % 3 == 0 )) && (( num % 5 != 0 )); then

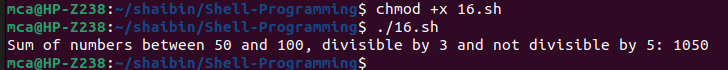
sum=$(( sum + num ))

fi

done

echo "Sum of numbers between 50 and 100, divisible by 3 and not divisible by 5: $sum"

**Output:**



**16. Write a Shell program to find the second highest number from a set of numbers.**

**Source Code:**

echo "Enter a list of numbers separated by spaces:"

read -a numbers

if [ ${#numbers[@]} -lt 2 ]; then

echo "Insufficient numbers provided. At least two numbers are required."

exit 1

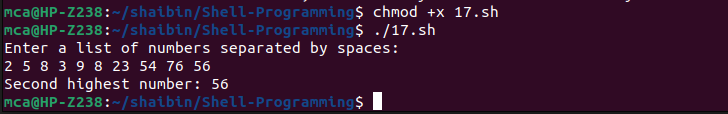
fi

sorted\_numbers=($(printf "%s\n" "${numbers[@]}" | sort -n -r))

second\_highest=${sorted\_numbers[1]}

echo "Second highest number: $second\_highest"

**Output:**



**17. Write a Shell program to find the sum of digits of a number using function.**

**Source Code:**

sum\_of\_digits() {

local num=$1

local sum=0

while [ $num -gt 0 ]; do

digit=$(( num % 10 ))

sum=$(( sum + digit ))

num=$(( num / 10 ))

done

echo $sum

}

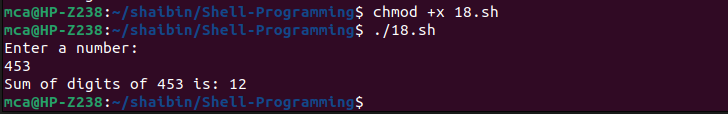
echo "Enter a number:"

read number

result=$(sum\_of\_digits $number)

echo "Sum of digits of $number is: $result"

**Output:**



**18. Write a Shell program to print the reverse of a number using function.**

**Source Code:**

echo "Enter a number:"

read num

reverse() {

local n=$1

local rev=0

while [ $n -gt 0 ]; do

remainder=$((n % 10))

rev=$((rev \* 10 + remainder))

n=$((n / 10))

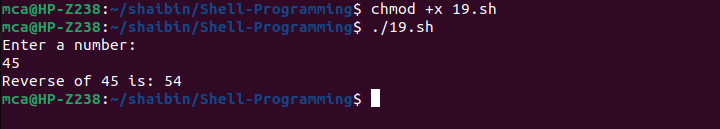
done

echo $rev

}

echo "Reverse of $num is: $(reverse $num)"

**Output:**



**19. Write a Shell program to find the factorial of a number using for loop.**

**Source Code:**

echo "Enter a number:"

read num

fact(){

fact=1

for((i=1; i<=num; i++)); do

fact=$((fact \* i))

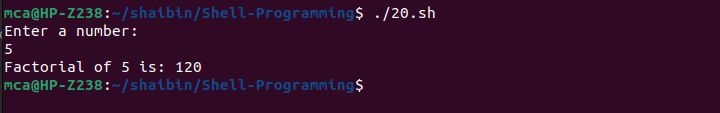
done

echo $fact

}

echo "Factorial of $num is: $(fact)"

**Output:**



**20. Write a Shell program to generate Fibonacci series.**

**Source Code:**

echo "Enter the number of terms:"

read n

a=0

b=1

echo "Fibonacci Series:"

for ((i=0; i<n; i++)); do

echo -n "$a "

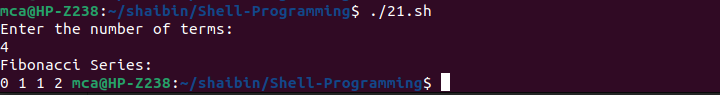
fn=$((a + b))

a=$b

b=$fn

done

**Output:**



**21.Write a shell script, which receives two filenames as arguments. It checks whether the two files contents are same or not. If they are same then second file is deleted.**

**Source Code:**

if [ $# -ne 2 ]; then

echo "Usage: $0 <file1> <file2>"

exit 1

fi

if cmp -s "$1" "$2"; then

echo "Files are the same. Deleting $2"

rm $2

else

echo "Files are different"

fi

**Output:**



**22. Write a Menu driven Shell script that Lists current directory, Prints Working Directory, displays Date and displays Users logged in.**

**Source Code:=**

PS3="Select option: "

select opt in "List current directory" "Print working directory" "Display date" "Display users logged in" "Exit"; do

case $opt in

"List current directory")

ls

;;

"Print working directory")

pwd

;;

"Display date")

date

;;

"Display users logged in")

who

;;

"Exit")

break

;;

\*)

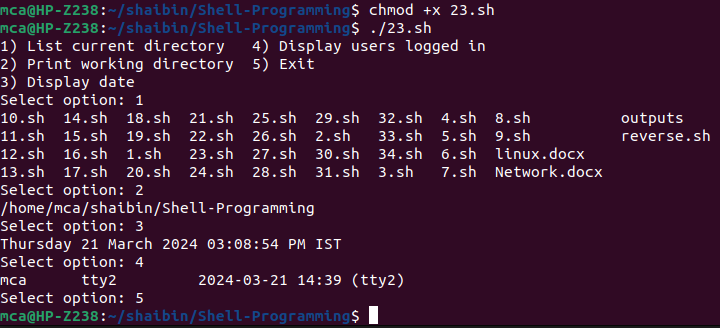
echo "Invalid option"

;;

esac

done

**Output:**



1. **Shell script to check executable rights for all files in the current directory, if a file does not have the execute permission then make it executable.**

**Source Code:**

#!/bin/bash

for file in \*; do

if [ -f $file ] && [ ! -x $file ]; then

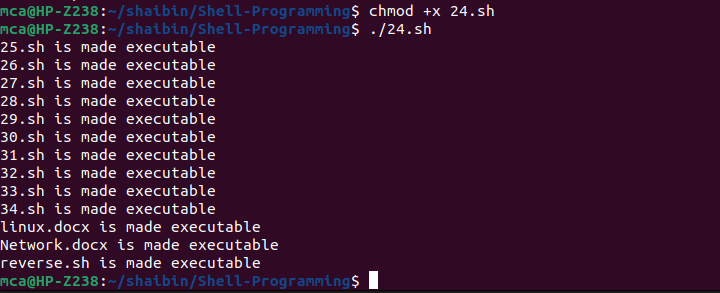
chmod +x $file

echo "$file is made executable"

fi

done

**Output:**



1. **Write a Shell program to generate all combinations of 1, 2, and 3 using loop.**

**Source Code:**

#!/bin/bash

for i in 1 2 3; do

for j in 1 2 3; do

for k in 1 2 3; do

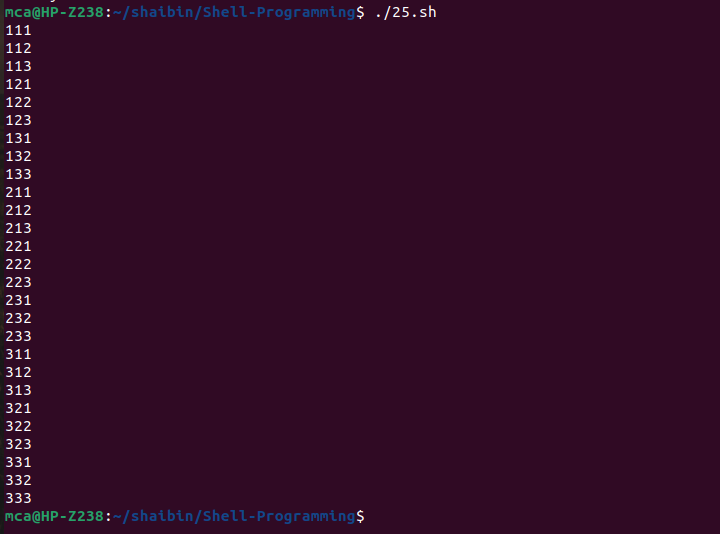
echo "$i$j$k"

done

done

done

**Output:**



1. **Write a Shell program to create the number series.**

**Source Code:**

#!/bin/bash

echo "Enter the number of terms:"

read n

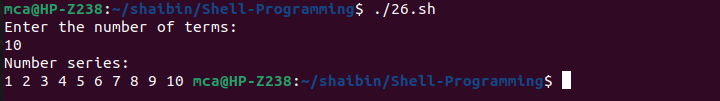
echo "Number series:"

for ((i=1; i<=n; i++)); do

echo -n "$i "

done

**Output:**



1. **Write a Shell program to create Pascal’s triangle.**

**Source Code:**

#!/bin/bash

echo "Enter the number of rows:"

read rows

for ((i=0; i<rows; i++)); do

for ((j=0; j<=i; j++)); do

if [ $j -eq 0 ] || [ $i -eq $j ]; then

coef=1

else

num=$((i-j+1))

den=$j

coef=$((coef \* num / den))

fi

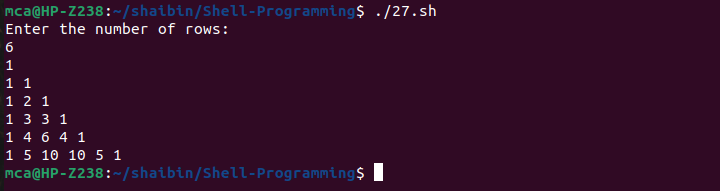
echo -n "$coef "

done

echo

done

**Output:**



1. **Write a Decimal to Binary Conversion Shell Script.**

**Source Code:**

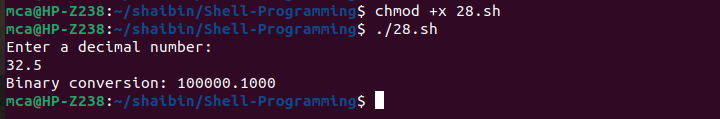
#!/bin/bash

echo "Enter a decimal number:"

read decimal

echo "Binary conversion: $(echo "obase=2; $decimal" | bc)"

**Output:**



1. **Write a Shell Script to Check Whether a String is Palindrome or not.**

**Source Code:**

#!/bin/bash

echo "Enter a string:"

read str

reverse=$(echo $str | rev)

if [ "$str" = "$reverse" ]; then

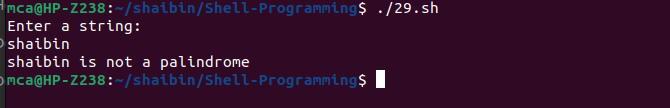
echo "$str is a palindrome"

else

echo "$str is not a palindrome"

fi

**Output:**



1. **Write a shell script to find out the unique words in a file and also count the occurrence of each of these words.**

**Source Code:**

#!/bin/bash

if [ $# -ne 1 ]; then

echo "Usage: $0 <filename>"

exit 1

fi

filename=$1

if [ ! -f "$filename" ]; then

echo "File '$filename' not found."

exit 1

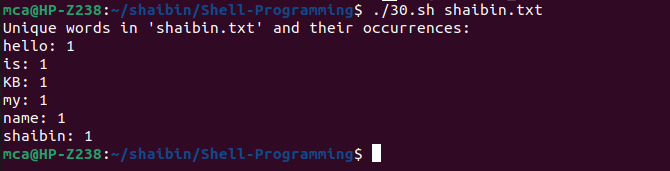
fi

unique\_words=$(tr -s '[:space:]' '\n' < "$filename" | sort | uniq -c | awk '{print $2 ": " $1}' | sort)

echo "Unique words in '$filename' and their occurrences:"

echo "$unique\_words"

**Output:**



1. **Write a shell script to get the total count of the word “Linux” in all the “.txt” files and also across files present in subdirectories.**

**Source Code:**

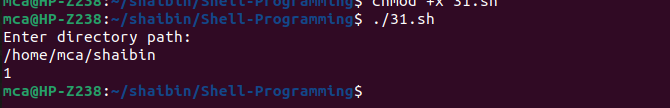
#!/bin/bash

echo "Enter directory path:"

read dir

grep -roh "Linux" $dir | wc -w

**Output:**



1. **Write a shell script to validate password strength. Here are a few assumptions for the password string. ( Length – minimum of 8 characters. Contain both alphabet and number. Include both the small and capital case letters.)**

**Source Code:**

#!/bin/bash

echo "Enter password:"

read password

if [[ ${#password} -lt 8 ]]; then

echo "Password length should be at least 8 characters"

exit 1

fi

if ! [[ $password =~ [0-9] ]]; then

echo "Password should contain at least one digit"

exit 1

fi

if ! [[ $password =~ [A-Z] ]]; then

echo "Password should contain at least one uppercase letter"

exit 1

fi

if ! [[ $password =~ [a-z] ]]; then

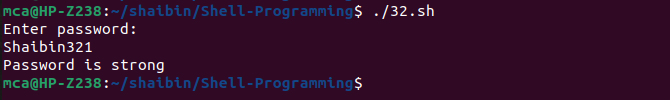
echo "Password should contain at least one lowercase letter"

exit 1

fi

echo "Password is strong"

**Output:**



1. **Write a shell script to print the count of files and subdirectories in the specified directory.**

**Source Code:**

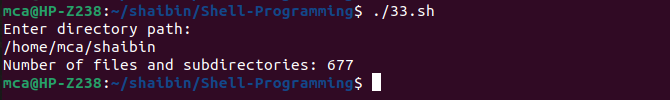
#!/bin/bash

echo "Enter directory path:"

read dir

echo "Number of files and subdirectories: $(find $dir -type d -or -type f | wc -l)"

**Output:**



1. **Write a shell script to reverse the list of strings and reverse each string further in the list.**

**Source Code:**

#!/bin/bash

reverse\_string() {

local string="$1"

local reversed=""

local len=${#string}

for (( i=len-1; i>=0; i-- )); do

reversed="$reversed${string:$i:1}"

done

echo "$reversed"

}

reverse\_list() {

local input\_list=("$@")

local reversed\_list=()

for (( i=${#input\_list[@]}-1; i>=0; i-- )); do

reversed\_list+=("$(reverse\_string "${input\_list[i]}")")

done

echo "${reversed\_list[@]}"

}

# Take user input for the list of strings

read -p "Enter the strings (separated by spaces): " -a input

# Call reverse\_list function with user input

reversed=$(reverse\_list "${input[@]}")

# Print the reversed list with reversed strings

echo "Reversed list with reversed strings:"

echo "$reversed"

**Output:**

