**Program 1**

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

echo -n "Enter a number:"

read n

echo -n "RESULT: "

if [ `expr $n % 2` == 0 ]

then

echo "$n is even"

else

echo "$n is Odd"

fi

**OUTPUT**



**Program 2**

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

echo "Enter the year:"

read year

x=`expr $year % 400`

y=`expr $year % 100`

z=`expr $year % 4`

if [ $x -eq 0 ] || [ $y -ne 0 ] && [ $z -eq 0 ]

then

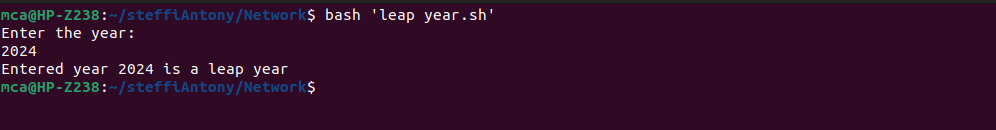
echo "Entered year $year is a leap year"

else

echo "Entered year $year is not a leap year"

fi

**OUTPUT**



**Program 3**

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

echo "Enter the radius of circle"

read r

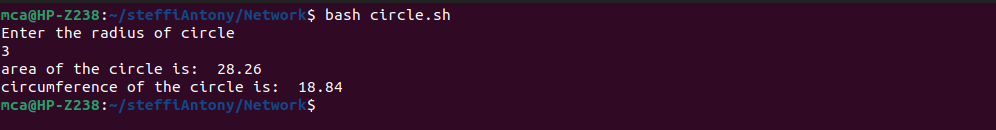
area=$(echo "3.14\*$r\*$r" | bc)

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

echo "area of the circle is: $area"

echo "circumference of the circle is: $circum"

**OUTPUT**



**Program 4**

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

echo "Enter a number:"

read number

reverse=$(echo $number | rev)

if [ $number -eq $reverse ]; then

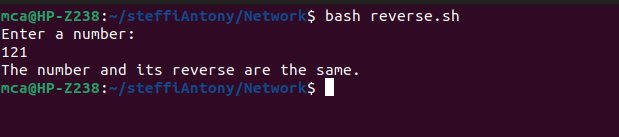
echo "The number and its reverse are the same."

else

echo "The number and its reverse are not the same."

Fi

**OUTPUT**



**Program 5**

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

echo "Enter a string:"

read string

reverse=$(echo $string | rev)

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

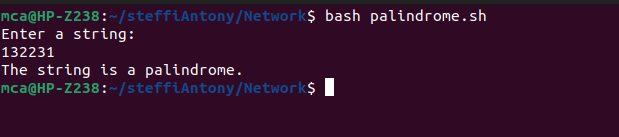
echo "The string is a palindrome."

else

echo "The string is not a palindrome."

fi

**OUTPUT**



**Program 6**

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

echo "Enter numbers separated by space:"

read -a numbers

sum\_even=0

sum\_odd=0

for num in "${numbers[@]}";

do

if [ $((num % 2)) -eq 0 ];

then

sum\_even=$((sum\_even + num))

else

sum\_odd=$((sum\_odd + num))

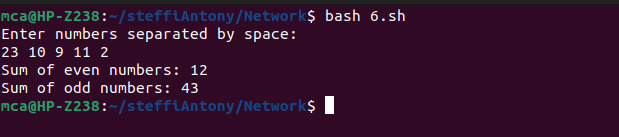
fi

done

echo "Sum of even numbers: $sum\_even"

echo "Sum of odd numbers: $sum\_odd"

**OUTPUT**



**Program 7**

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

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)

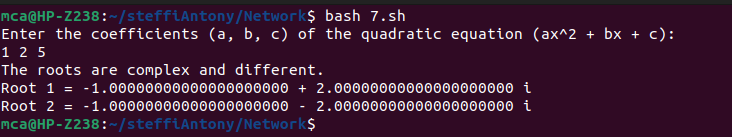
echo "The roots are complex and different."

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

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

fi

**OUTPUT**



**Program 8**

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

echo "Enter a number:"

read number

length=${#number}

sum=0

for ((i=0; i<$length; i++)); do

digit=${number:i:1}

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

done

if [ $sum -eq $number ]; then

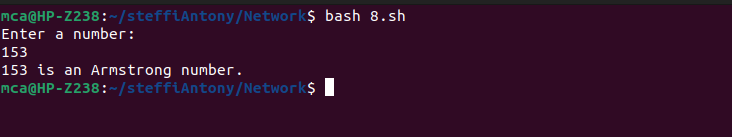
echo "$number is an Armstrong number."

else

echo "$number is not an Armstrong number."

Fi

**OUTPUT**



**Program 9**

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

echo "Enter a number:"

read number

is\_prime=true

if [ $number -lt 2 ]; then

is\_prime=false

fi

for ((i=2; i<=number/2; i++)); do

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

is\_prime=false

break

fi

done

if $is\_prime; then

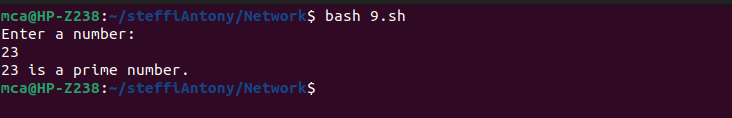
echo "$number is a prime number."

else

echo "$number is not a prime number."

Fi

**OUTPUT**



**Program 10**

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

echo "Prime numbers between 1 and 50 are:"

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

is\_prime=true

for ((j=2; j<=i/2; j++)); do

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

is\_prime=false

break

fi

done

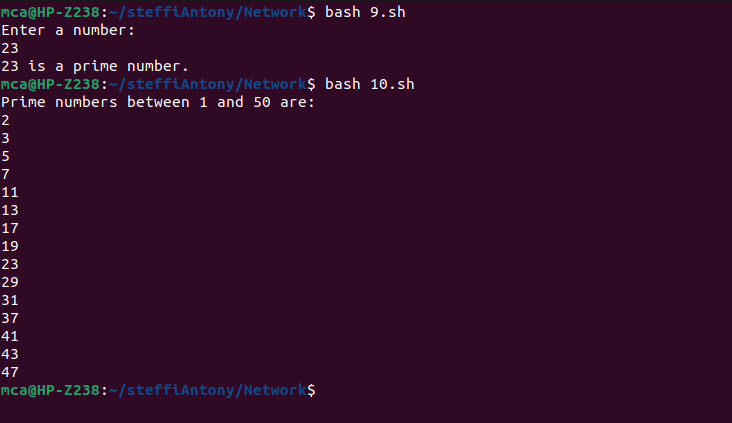
if $is\_prime; then

echo $i

fi

done

**OUTPUT**



**Program 11**

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

echo "Enter a number:"

read num

sum=0

while [ $num -gt 0 ]; do

digit=$(( $num % 10 ))

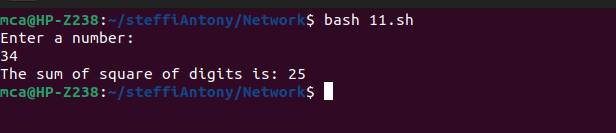
sum=$(( $sum + $digit \* $digit ))

num=$(( $num / 10 ))

done

echo "The sum of square of digits is: $sum"

**OUTPUT**



**Program 12**

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

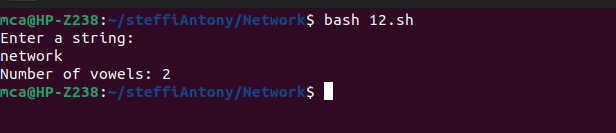
echo "Enter a string:"

read str

count=$(echo $str | grep -o -i "[aeiou]" | wc -l)

echo "Number of vowels: $count"

**OUTPUT**



**Program 13**

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

calculate\_grade() {

if [ $1 -ge 90 ]; then

grade="A"

elif [ $1 -ge 80 ]; then

grade="B"

elif [ $1 -ge 70 ]; then

grade="C"

elif [ $1 -ge 60 ]; then

grade="D"

else

grade="F"

fi

echo $grade

}

echo "Enter student name:"

read name

echo "Enter student's mark:"

read mark

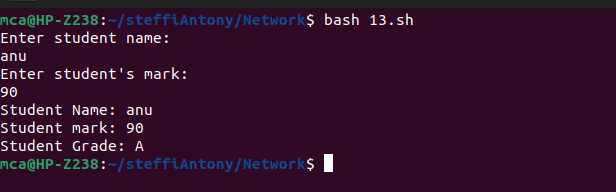
grade=$(calculate\_grade $mark)

echo "Student Name: $name"

echo "Student mark: $mark"

echo "Student Grade: $grade"

**OUTPUT**



**Program 14**

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

echo "Enter numbers separated by space:"

read -a numbers

largest=${numbers[0]}

smallest=${numbers[0]}

for num in "${numbers[@]}"; do

if [ $num -gt $largest ]; then

largest=$num

fi

if [ $num -lt $smallest ]; then

smallest=$num

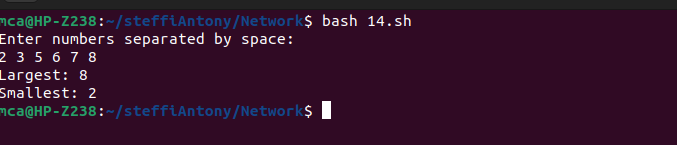
fi

done

echo "Largest: $largest"

echo "Smallest: $smallest"

**OUTPUT**



**Program 15**

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

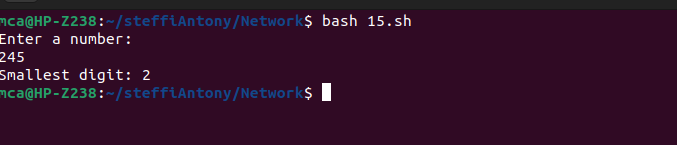
echo "Enter a number:"

read num

smallest=$(echo $num | grep -o "[0-9]" | sort | head -n1)

echo "Smallest digit: $smallest"

**OUTPUT**



**Program 16**

**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.**

sum=0

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

if [ $((i % 3)) -eq 0 ] && [ $((i % 5)) -ne 0 ]; then

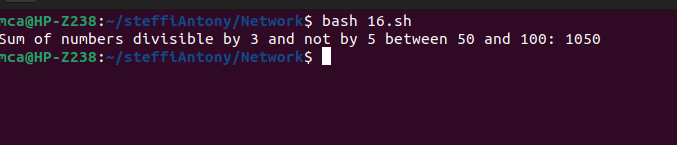
sum=$((sum + i))

fi

done

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

**OUTPUT**



**Program 17**

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

echo "Enter numbers separated by space:"

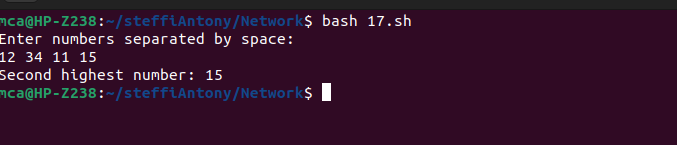
read -a numbers

IFS=$'\n' sorted=($(sort -n <<<"${numbers[\*]}"))

len=${#sorted[@]}

echo "Second highest number: ${sorted[len-2]}"

**OUTPUT**



**Program 18**

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

echo "Enter a number:"

read num

sum\_digits() {

local n=$1

local sum=0

while [ $n -gt 0 ]; do

sum=$((sum + n % 10))

n=$((n / 10))

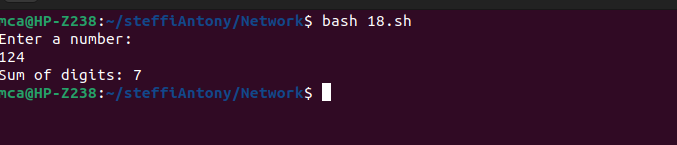
done

echo $sum

}

echo "Sum of digits: $(sum\_digits $num)"

**OUTPUT**



**Program 19**

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

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))

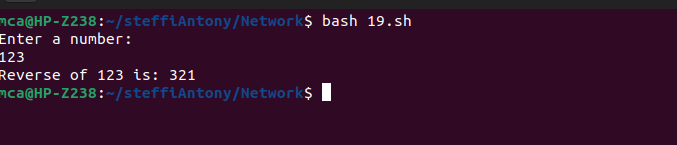
done

echo $rev

}

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

**OUTPUT**



**Program 20**

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

echo "Enter a number:"

read num

fact(){

fact=1

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

fact=$((fact \* i))

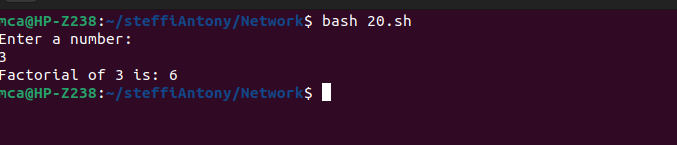
done

echo $fact

}

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

**OUTPUT**



**Program 21**

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

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 "

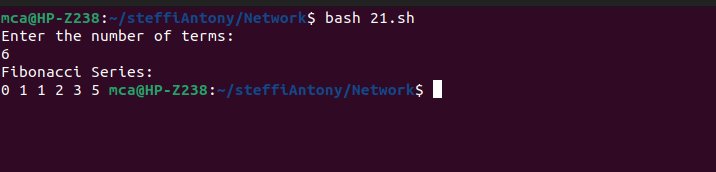
fn=$((a + b))

a=$b

b=$fn

done

**OUTPUT**



**Program 22**

**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**.

if [ $# -ne 2 ]; then

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

exit 0

fi

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

echo "Files are the same. Deleting $2"

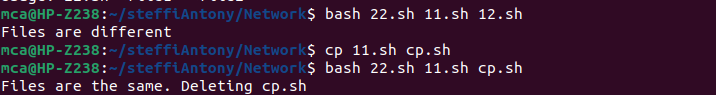
rm $2

else

echo "Files are different"

fi

**OUTPUT**



**Program 23**

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

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

;;

\*)

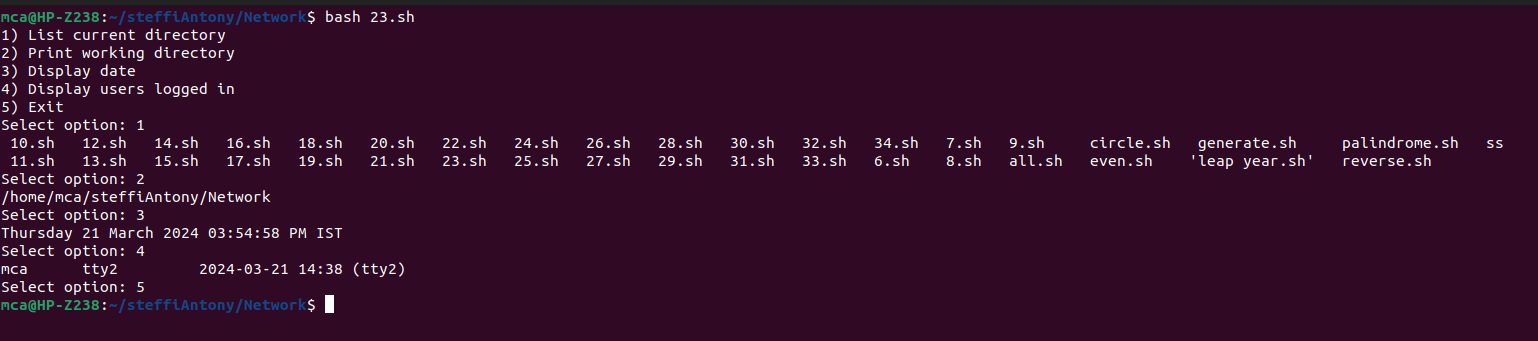
echo "Invalid option"

;;

esac

done

**OUTPUT**



**Program 24**

**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.**

for file in \*; do

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

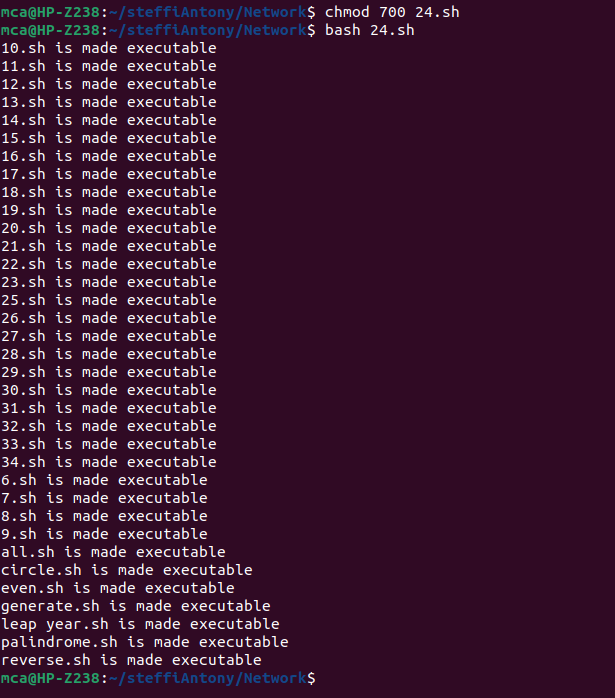
chmod u+rwx "$file"

echo "$file is made executable"

fi

done

**OUTPUT**



**Program 25**

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

for i in 1 2 3; do

for j in 1 2 3; do

for k in 1 2 3; do

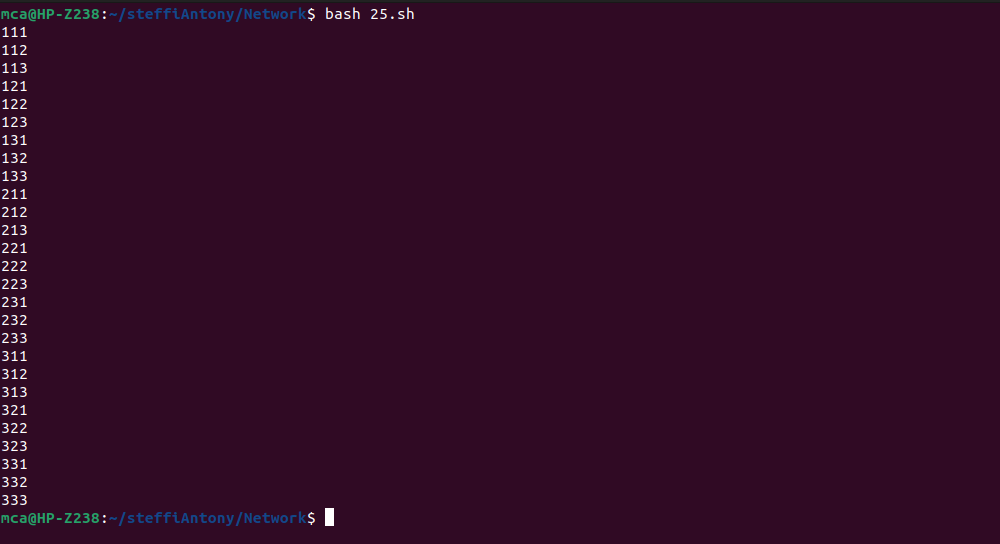
echo "$i$j$k"

done

done

done

**OUTPUT**



**Program 26**

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

echo "Enter the number of terms:"

read n

echo "Number series:"

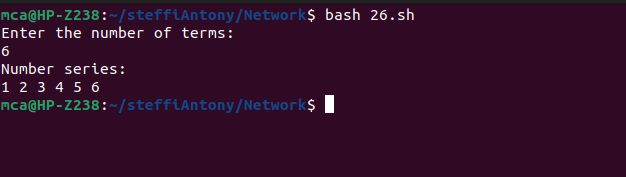
for ((i=1; i<=n; i++)); do

echo -n "$i "

done

echo ""

**OUTPUT**

****

**Program 27**

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

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))

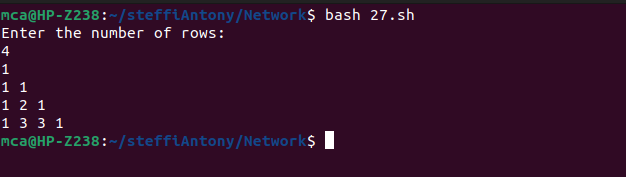
fi

echo -n "$coef "

done

echo

done

**OUTPUT**

**Program 28**

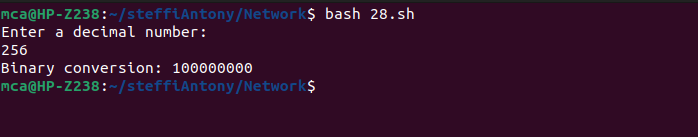
**Write a Decimal to Binary Conversion Shell Script.**

echo "Enter a decimal number:"

read decimal

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

**OUTPUT**



**Program 29**

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

echo "Enter a string:"

read str

reverse=$(echo $str | rev)

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

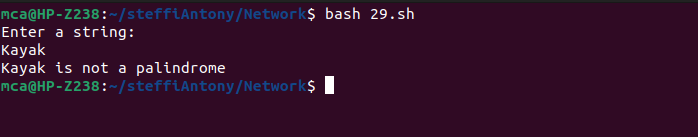
echo "$str is a palindrome"

else

echo "$str is not a palindrome"

fi

**OUTPUT**



**Program 30**

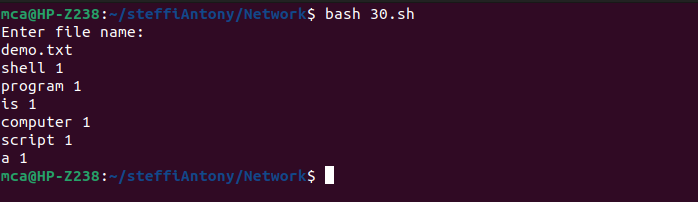
**Write a shell script to find out the unique words in a file and also count the occurrence of each of these words**.

echo "Enter file name:"

read filename

awk '{for(i=1;i<=NF;i++) a[$i]++} END {for(k in a) print k, a[k]}' $filename

**OUTPUT**



**Program 31**

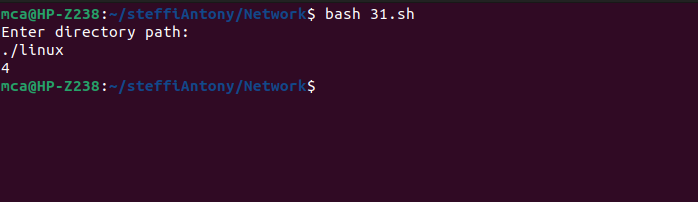
**Write a shell script to get the total count of the word “Linux” in all the “.txt” files and also across files present in subdirectories.**

echo "Enter directory path:"

read dir

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

**OUTPUT**



**Program 32**

**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.**

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

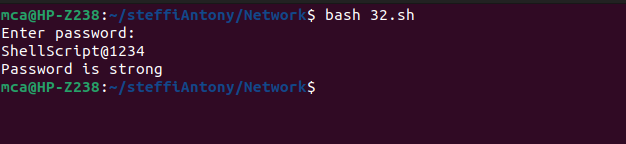
echo "Password should contain at least one lowercase letter"

exit 1

fi

echo "Password is strong"

**OUTPUT**



**Program 33**

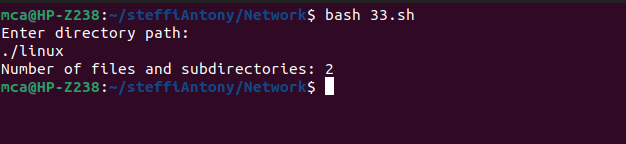
**Write a shell script to print the count of files and subdirectories in the specified directory.**

echo "Enter directory path:"

read dir

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

**OUTPUT**



**Program 34**

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

echo "Enter strings separated by space:"

read -a strings

leng=${#strings[@]}

echo $leng

reversed\_strings=()

for ((i=($leng-1); i>=0; i--)); do

reversed\_strings+=(${strings[i]})

done

echo "Initial array : ${strings[@]}"

echo -n "Reversed list of strings: "

for string in "${reversed\_strings[@]}"; do

echo -n "$(echo $string | rev) "

done; echo

**OUTPUT**