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

**CODE**

function check\_odd\_even {

if [ $(( $1 % 2 )) -eq 0 ]; then

echo "$1 is even"

else

echo "$1 is odd"

fi

}

echo "Enter a number:"

read num

check\_odd\_even $num



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

**CODE**

echo -n "Enter year: "

read year

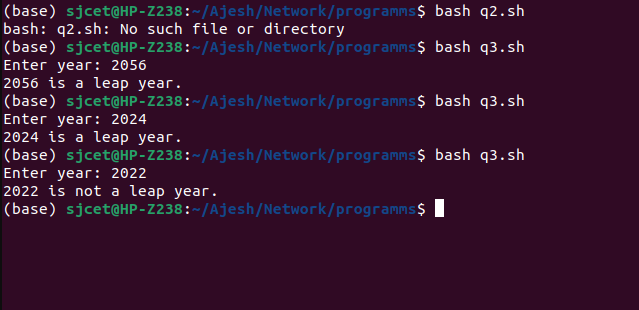
if [ $((year % 4)) -eq 0 ] && [ $((year % 100)) -ne 0 ] || [ $((year % 400)) -eq 0 ]; then

echo "$year is a leap year."

else

echo "$year is not a leap year."

fi



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

**CODE**

echo "Enter the radius of the circle: "

read radius

# Calculate the area of the circle

area=$(echo "scale=2; 3.14 \* ($radius^2)" | bc)

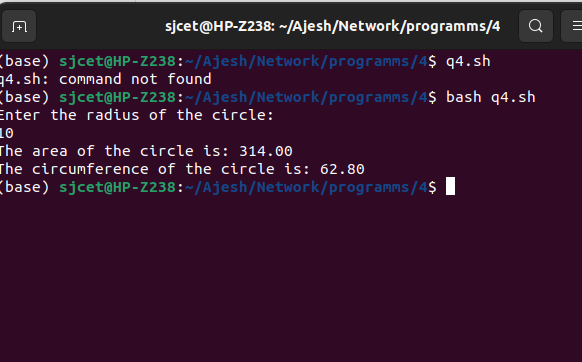
# Calculate the circumference of the circle

circumference=$(echo "scale=2; 2 \* 3.14 \* $radius" | bc)

# Print the results

echo "The area of the circle is: $area"

echo "The circumference of the circle is: $circumference"



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

**CODE**

echo "Enter a number: "

read number

# Reverse the number

reverse=$(echo $number | rev)

# Check if the number and its reverse are the same

if [ "$number" -eq "$reverse" ]

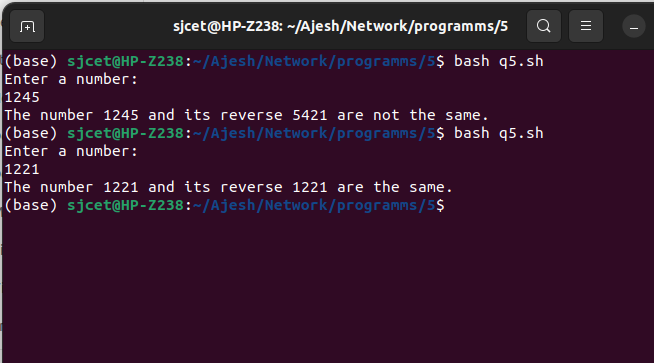
then

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

else

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

fi



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

**CODE**

echo "Enter a string: "

read string

# Reverse the string

reverse=$(echo "$string" | rev)

# Check if the string and its reverse are the same

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

then

echo "The string \"$string\" is a palindrome."

else

echo "The string \"$string\" is not a palindrome."

fi



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

**CODE**

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

read -a numbers

# Initialize variables to store the sum of even and odd numbers

even\_sum=0

odd\_sum=0

# Loop through the numbers and add them to the appropriate sum

for number in "${numbers[@]}"

do

if [ $((number % 2)) -eq 0 ]

then

even\_sum=$((even\_sum + number))

else

odd\_sum=$((odd\_sum + number))

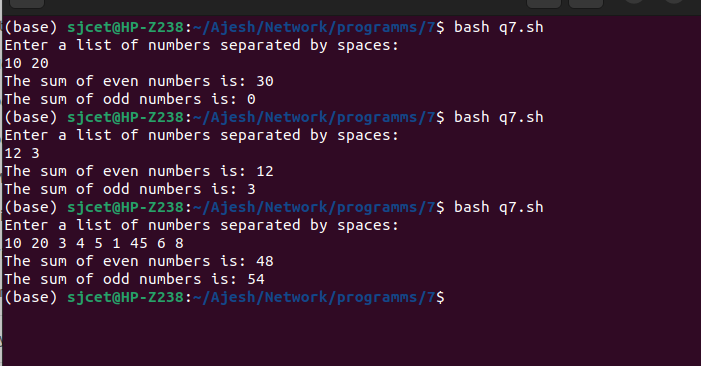
fi

done

# Print the results

echo "The sum of even numbers is: $even\_sum"

echo "The sum of odd numbers is: $odd\_sum"



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

**CODE**

echo "Enter the coefficients of the quadratic equation (a, b, c): "

read a b c

# Calculate the discriminant

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

# Check if the discriminant is negative (no real roots)

if [ $discriminant -lt 0 ]

then

echo "The quadratic equation has no real roots."

else

# Calculate the roots

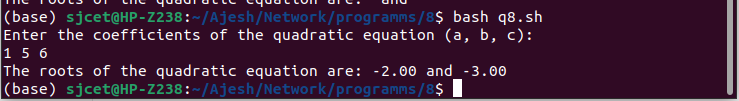
root1=$(echo "scale=2; (-$b + sqrt($discriminant)) / (2\*$a)" | bc)

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

# Print the roots

echo "The roots of the quadratic equation are: $root1 and $root2"

fi



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

**CODE**

echo "Enter an integer: "

read number

# Count the number of digits in the number

count=${#number}

# Initialize the sum to 0

sum=0

# Loop through the digits of the number an+

d calculate the sum

for (( i=0; i<count; i++ ))

do

digit=${number:i:1}

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

done

# Check if the number is an Armstrong number

if [ "$sum" -eq "$number" ]

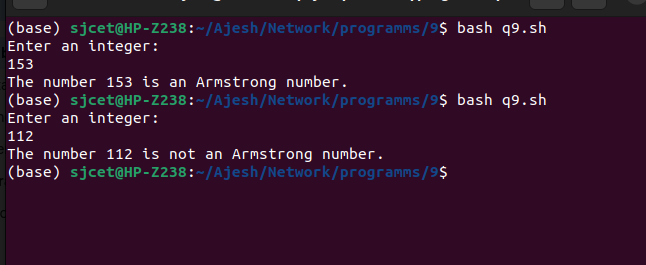
then

echo "The number $number is an Armstrong number."

else

echo "The number $number is not an Armstrong number."

fi



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

**CODE**

echo "Enter an integer: "

read number

# Initialize the flag variable to 1

flag=1

# Check if the number is prime

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

do

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

then

flag=0

break

fi

done

# Output the result

if [ $number -eq 1 ]

then

echo "1 is neither prime nor composite."

elif [ $flag -eq 1 ]

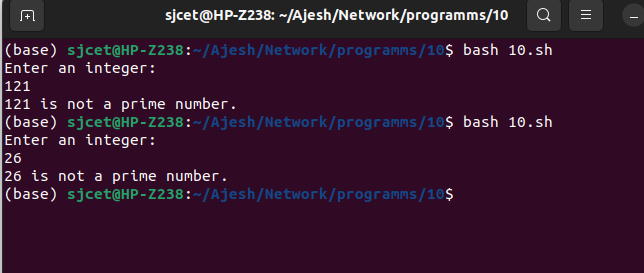
then

echo "$number is a prime number."

else

echo "$number is not a prime number."

fi



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

**CODE**

echo "Prime numbers between 1 and 50 are:"

# Check each number between 1 and 50 for primality

for (( number=2; number<=50; number++ ))

do

flag=1

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

do

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

then

flag=0

break

fi

done

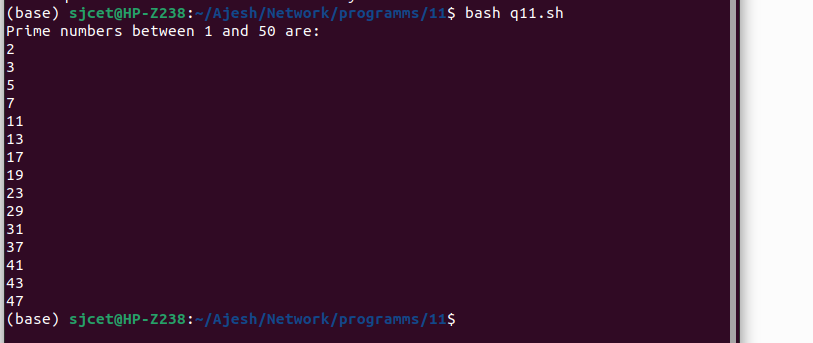
if [ $flag -eq 1 ]

then

echo $number

fi

done



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

**CODE**

echo "Enter a number: "

read number

# Initialize the sum to 0

sum=0

# Loop through the digits of the number and calculate the sum of their squares

while [ $number -ne 0 ]

do

digit=$((number % 10))

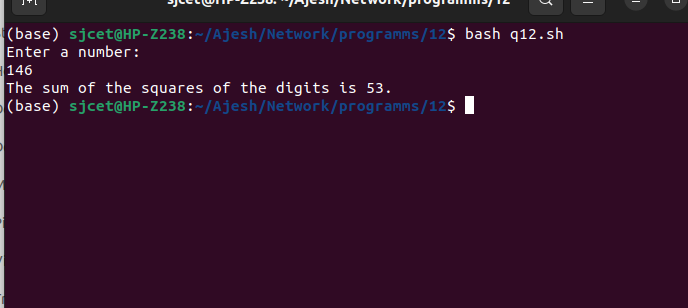
sum=$((sum + digit \* digit))

number=$((number / 10))

done

# Output the result

echo "The sum of the squares of the digits is $sum."



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

**CODE**

echo "Enter a line of text: "

read line

# Initialize the vowel count to 0

count=0

# Loop through each character of the line and check if it is a vowel

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

do

char=${line:$i:1}

if [[ $char == [aeiouAEIOU] ]]

then

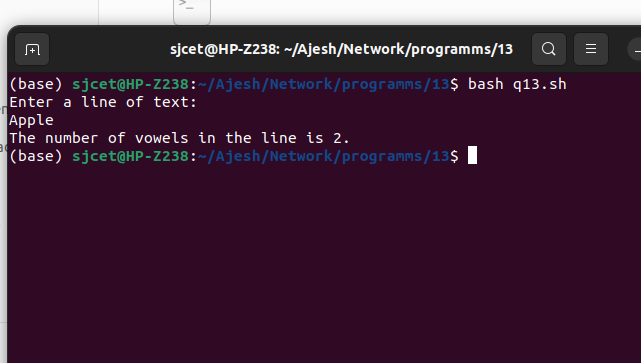
count=$((count + 1))

fi

done

# Output the result

echo "The number of vowels in the line is $count."



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

**CODE**

declare -A grades=(

[Alice]=90

[Bob]=80

[Charlie]=70

[David]=60

[Emma]=50

)

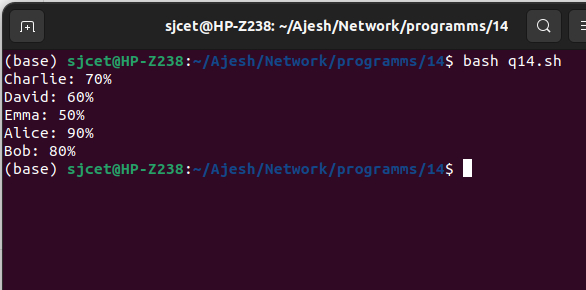
# Loop through the student names and output their grades

for name in "${!grades[@]}"

do

echo "$name: ${grades[$name]}%"

Done



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

**CODE**

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

read numbers

# Convert the input string to an array of numbers

IFS=' ' read -ra nums <<< "$numbers"

# Initialize the min and max variables to the first number in the array

min=${nums[0]}

max=${nums[0]}

# Loop through the remaining numbers in the array and update min and max as needed

for num in "${nums[@]}"

do

if (( num < min )); then

min=$num

fi

if (( num > max )); then

max=$num

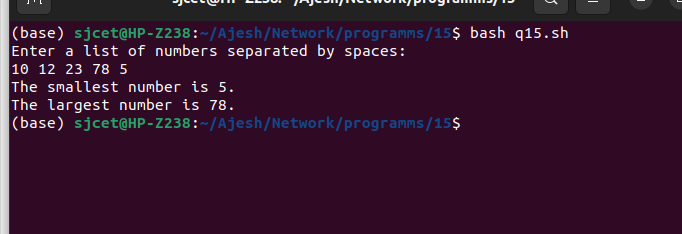
fi

done

# Output the result

echo "The smallest number is $min."

echo "The largest number is $max."



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

**CODE**

echo "Enter a number: "

read num

# Initialize the min variable to the first digit of the number

min=${num:0:1}

# Loop through the remaining digits of the number and update min as needed

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

do

digit=${num:$i:1}

if (( digit < min )); then

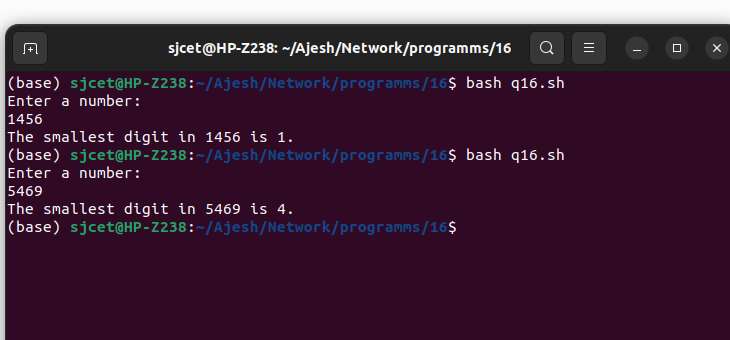
min=$digit

fi

done

# Output the result

echo "The smallest digit in $num is $min."



1. **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.**

**CODE**

sum=0

# Loop through the numbers between 50 and 100

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

do

# Check if the number is divisible by 3 and not divisible by 5

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

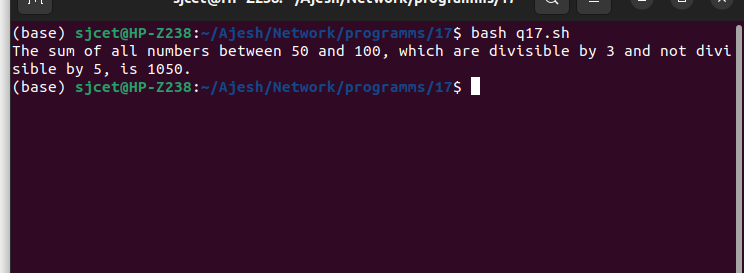
sum=$((sum + num))

fi

done

# Output the result

echo "The sum of all numbers between 50 and 100, which are divisible by 3 and not divisible by 5, is $sum."



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

**CODE**

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

read numbers

# Convert the space-separated string to an array

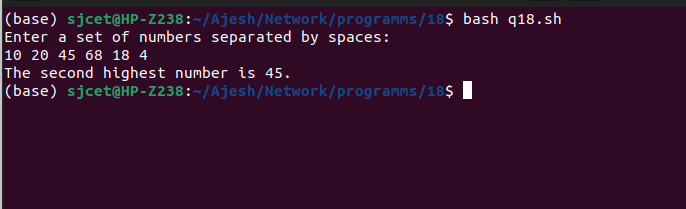
arr=($numbers)

# Sort the array in descending order

sorted\_arr=($(echo "${arr[@]}" | tr " " "\n" | sort -rn))

# Output the second highest number

echo "The second highest number is ${sorted\_arr[1]}."



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

**CODE**

# Define the function to calculate the sum of digits

sum\_of\_digits() {

num=$1

sum=0

while [ $num -gt 0 ]

do

digit=$((num % 10))

sum=$((sum + digit))

num=$((num / 10))

done

echo $sum

}

# Prompt the user to enter a number

echo "Enter a number: "

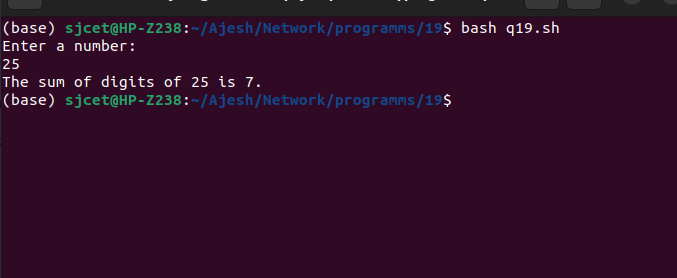
read num

# Call the function to calculate the sum of digits

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

# Output the result

echo "The sum of digits of $num is $result."



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

**CODE**

# Define the function to reverse a number

reverse\_number() {

num=$1

rev=0

while [ $num -gt 0 ]

do

digit=$((num % 10))

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

num=$((num / 10))

done

echo $rev

}

# Prompt the user to enter a number

echo "Enter a number: "

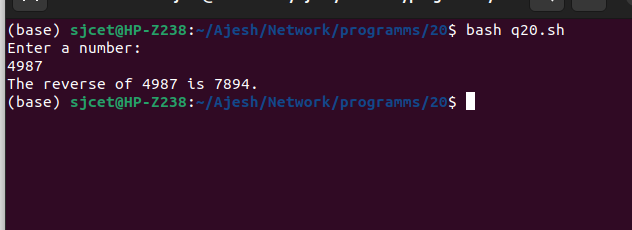
read num

# Call the function to reverse the number

result=$(reverse\_number $num)

# Output the result

echo "The reverse of $num is $result."



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

**CODE**

# Prompt the user to enter a number

echo "Enter a number: "

read num

# Initialize the factorial to 1

factorial=1

# Calculate the factorial using a for loop

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

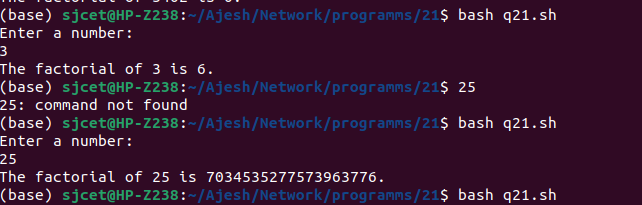
do

factorial=$((factorial \* i))

done

# Output the result

echo "The factorial of $num is $factorial."



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

**CODE**

# Prompt the user to enter the number of terms to generate

echo "Enter the number of terms to generate: "

read num

# Initialize the first two terms of the series

a=0

b=1

# Output the first two terms

echo -n "$a $b"

# Generate the rest of the series using a loop

for (( i=3; i<=$num; i++ ))

do

# Calculate the next term

c=$((a + b))

# Output the next term

echo -n " $c"

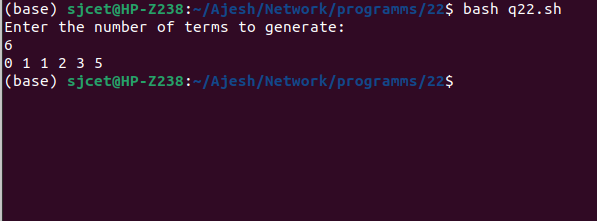
# Shift the values of a and b to prepare for the next iteration

a=$b

b=$c

done

Echo



1. **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.**

**CODE**

if [ $# -ne 2 ]; then

echo "Usage: $0 file1 file2"

exit 1

fi

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

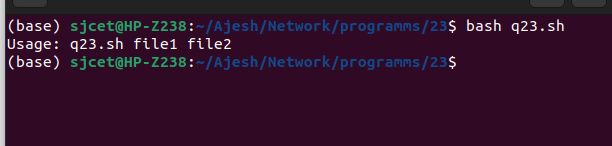
echo "The contents of $1 and $2 are the same."

rm "$2"

else

echo "The contents of $1 and $2 are different."

fi



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

**CODE**

echo "Select an option:"

echo "1. List current directory"

echo "2. Print working directory"

echo "3. Display date"

echo "4. Display users logged in"

read option

case $option in

1)

ls -l

;;

2)

pwd

;;

3)

date

;;

4)

who

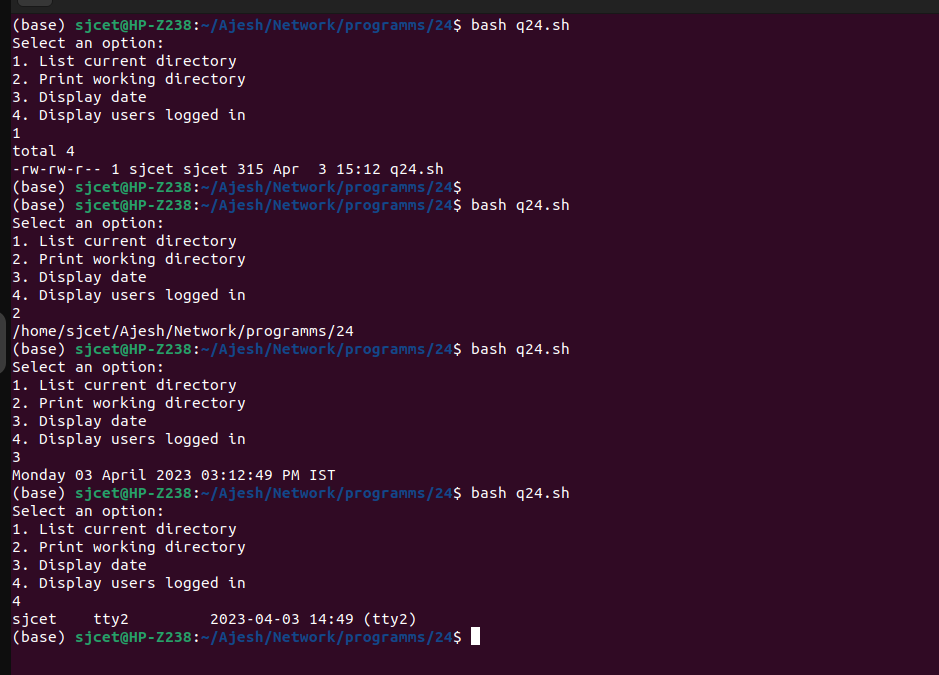
;;

\*)

echo "Invalid option selected"

;;

Esac



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.**

**CODE**

find . -type f | while read file; do

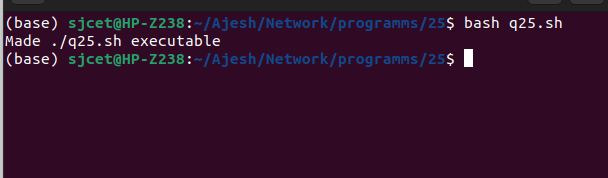
if [ ! -x "$file" ]; then

chmod +x "$file"

echo "Made $file executable"

fi

done



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

**CODE**

or i in 1 2 3; do

for j in 1 2 3; do

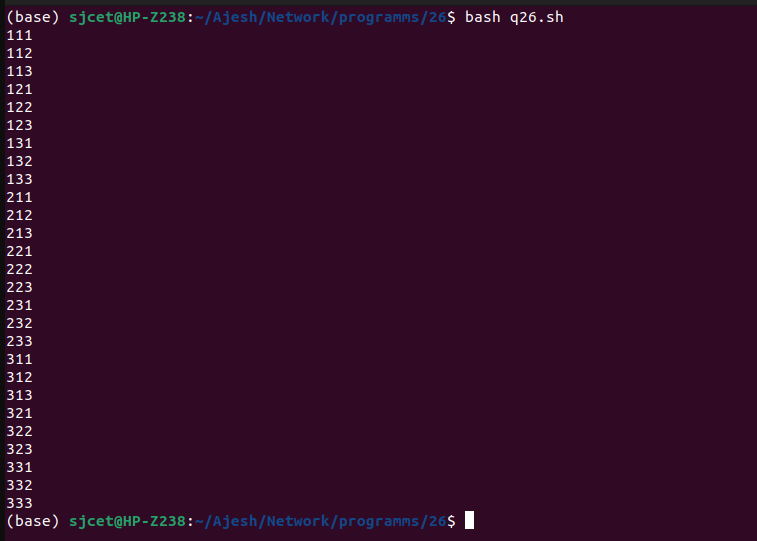
for k in 1 2 3; do

echo "$i$j$k"

done

done

done



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

1

2 3

4 5 6

7 8 9 10

**CODE**

count=1

for (( i=1; i<=4; i++ ))

do

for (( j=1; j<=i; j++ ))

do

echo -n "$count "

count=$((count+1))

done

echo ""

done



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

**CODE**

#!/bin/bash

# set the number of rows

echo "Enter the number of rows to generate for Pascal's triangle:"

read rows

# initialize the first row

row=1

echo $row

# loop over the remaining rows

for ((i=1; i<$rows; i++)); do

# initialize the row with the left-most element

prev\_row=($row)

row=${prev\_row[0]}

# loop over the remaining elements in the row

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

# calculate the current element

current=$((prev\_row[j-1] + prev\_row[j]))

# append the current element to the row

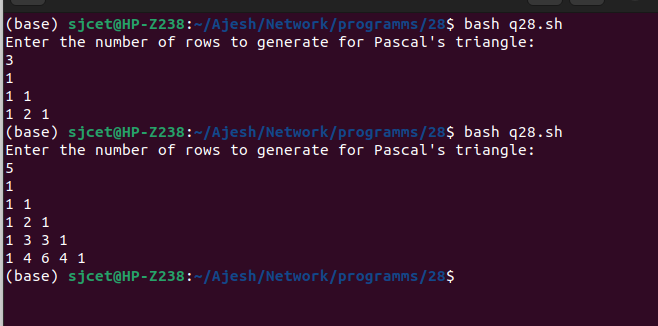
row="$row $current"

done

# print the row

echo $row

done



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

**CODE**

#!/bin/bash

# Prompt the user for the decimal number to convert

echo "Enter a decimal number: "

read decimal

# Convert the decimal number to binary

binary=""

while [ $decimal -gt 0 ]; do

remainder=$((decimal % 2))

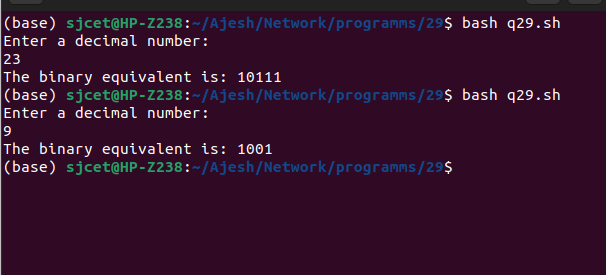
binary="$remainder$binary"

decimal=$((decimal / 2))

done

# Print the binary number

echo "The binary equivalent is: $binary"



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

**CODE**

#!/bin/bash

# Prompt the user for the string to check

echo "Enter a string: "

read string

# Reverse the string

reverse=$(echo $string | rev)

# Check if the string is equal to its reverse

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

echo "$string is a palindrome."

else

echo "$string is not a palindrome."

Fi



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

**CODE**

#!/bin/bash

# Prompt the user for the file name

echo "Enter the file name: "

read file

# Check if the file exists

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

echo "File not found."

exit 1

fi

# Convert the contents of the file to lowercase and replace all non-alphanumeric characters with spaces

contents=$(tr '[:upper:]' '[:lower:]' < $file | sed 's/[^a-z0-9]/ /g')

# Create an array of words from the file contents

words=($contents)

# Loop through the array of words and count their occurrences

declare -A count

for word in "${words[@]}"; do

if [ -n "$word" ]; then

((count[$word]++))

fi

done

# Print the unique words and their counts

echo "Unique words in $file:"

for word in "${!count[@]}"; do

echo "$word: ${count[$word]}"

done



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.**

**CODE**

#!/bin/bash

# Set the search directory

search\_dir="."

# Find all ".txt" files in the search directory and its subdirectories

files=$(find "$search\_dir" -type f -name "\*.txt")

# Initialize the count

count=0

# Loop through each file and count the occurrences of "Linux"

for file in $files; do

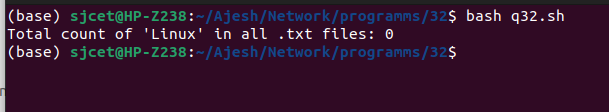
occurrences=$(grep -o "Linux" "$file" | wc -l)

count=$((count + occurrences))

done

# Print the total count

echo "Total count of 'Linux' in all .txt files: $count"



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.**

**CODE**

#!/bin/bash

# Prompt user to enter password

echo "Please enter a password: "

read password

# Check if password length is at least 8 characters

if [ ${#password} -lt 8 ]

then

echo "Password must have a minimum of 8 characters."

exit 1

fi

# Check if password contains both alphabet and number

if ! [[ "$password" =~ [[:alpha:]] && "$password" =~ [[:digit:]] ]]

then

echo "Password must contain both alphabet and number."

exit 1

fi

# Check if password includes both small and capital case letters

if ! [[ "$password" =~ [[:lower:]] && "$password" =~ [[:upper:]] ]]

then

echo "Password must include both small and capital case letters."

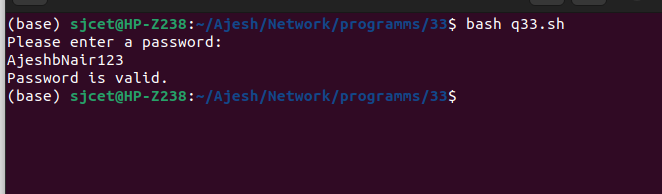
exit 1

fi

# If all checks pass, password is valid

echo "Password is valid."

exit 0



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

**CODE**

#!/bin/bash

if [ $# -eq 0 ]; then

echo "Usage: $0 directory"

exit 1

fi

directory=$1

if [ ! -d $directory ]; then

echo "Error: $directory is not a directory"

exit 1

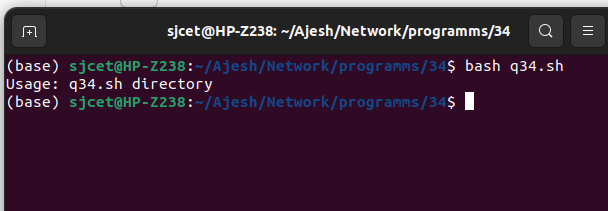
fi

num\_files=$(find $directory -maxdepth 1 -type f | wc -l)

num\_dirs=$(find $directory -maxdepth 1 -type d | wc -l)

echo "Number of files in $directory: $num\_files"

echo "Number of directories in $directory: $((num\_dirs - 1))"



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

**CODE**

#!/bin/bash

# prompt user for list of strings

read -p "Enter a list of strings separated by spaces: " string\_list

# split string\_list into an array of strings

read -a strings <<< "$string\_list"

# reverse the array of strings

for (( i=${#strings[@]}-1; i>=0; i-- ))

do

# reverse each string in the array

reversed\_string=""

for (( j=${#strings[i]}-1; j>=0; j-- ))

do

reversed\_string="$reversed\_string${strings[i]:$j:1}"

done

# replace the original string with the reversed string

strings[i]=$reversed\_string

done

# print the reversed and reversed strings

echo "${strings[@]}"

