1. Practice Basic Shell Commands like:- ls, cd, du, pwd, man, cat, more, less, head, tail, mkdir, cp, mv, rm, touch, grep, sort, wc, cut, echo…
2. Write a Shell program to check if the given number is even or odd.

read -p "enter number : " n

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

then

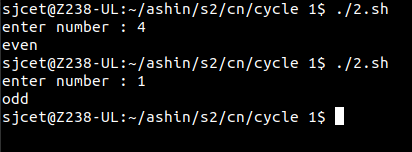
echo "even "

else

echo "odd"

fi

Output



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

read -p "enter year : " y

if [ `expr $y % 4` == 0 ]

then

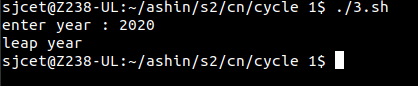
echo "leap year"

else

echo "not"

fi

Output



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

read -p "Enter the radius of the circle" r

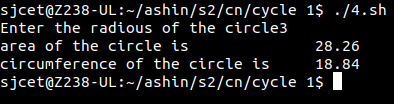
area=$(echo "3.14\*$r\*$r" | bc )

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

echo "area of the circle is " $area

echo "circumference of the circle is " $circum

Output



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

echo "Enter a Number:"

read a

rev=0

sd=0

or=$a

while [ $a -gt 0 ]

do

sd=`expr $a % 10`

temp=`expr $rev \\* 10`

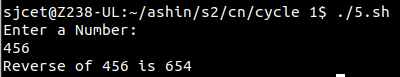
rev=`expr $temp + $sd`

a=`expr $a / 10`

done

echo "Reverse of $or is $rev"

Output



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

echo "Enter a String"

read input

reverse=""

len=${#input}

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

do

reverse="$reverse${input:$i:1}"

done

if [ $input == $reverse ]

then

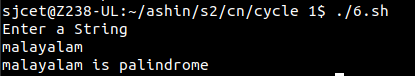
echo "$input is palindrome"

else

echo "$input is not palindrome"

fi

Output



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

echo "enter"

read num

rev=0

even=0

odd=0

while [ $num -gt 0 ]

do

tmp=$(( $num % 10 ))

if(( $tmp % 2 == 0 ))

then

even=$(( $even + $tmp ))

else

odd=$(( $odd + $tmp ))

fi

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

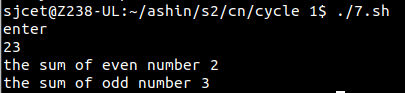
num=$(( $num / 10 ))

done

echo the sum of even number $even

echo the sum of odd number $odd

Output



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

echo Enter the coefficient of x^2:

read a

echo Enter the coefficient of x:

read b

echo Enter the constant term:

read c

f=`echo "-($b)" |bc`

p=`expr 2 \\* $a`

if [ $a -ne 0 ]

then

d=`echo \( \( $b \\* $b \) - \( 4 \\* $a \\* $c \) \) | bc`

if [ $d -lt 0 ]

then

x=`echo "-($d)" | bc`

s=`echo "scale=2; sqrt ( $x )" | bc`

echo The first root is:

echo "($f + $s i) / $p"

echo The second root is:

echo "($f - $s i) / $p"

elif [ $d -eq 0 ]

then

res=`expr $f / $p`

echo The root is: $res

else

s=`echo "scale=2; sqrt( $d )" | bc`

res1=`echo "scale=2; ( $f + $s) / ( $p )"|bc`

res2=`echo "scale=2; ( $f - $s) / ( $p )"|bc`

echo The first root is: $res1

echo The second root is: $res2

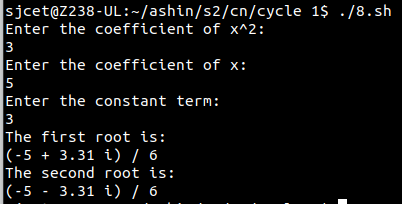
fi

else

echo Coefficient of x^2 can not be 0.

fi

Output



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

echo "Enter a number: "

read c

x=$c

sum=0

r=0

n=0

while [ $x -gt 0 ]

do

r=`expr $x % 10`

n=`expr $r \\* $r \\* $r \\* $r`

sum=`expr $sum + $n`

x=`expr $x / 10`

done

if [ $sum -eq $c ]

then

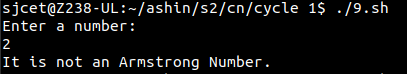
echo "It is an Armstrong Number."

else

echo "It is not an Armstrong Number."

fi

Output



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

echo -e "Enter Number : \c"

read n

while [ $n -gt 2 ]

do

for((i=2; i<=$n/2; i++))

do

ans=$(( n%i ))

if [ $ans -eq 0 ]

then

echo "$n is not a prime number."

exit 0

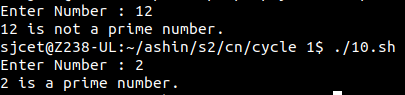
fi

done

done

echo "$n is a prime number."

Output



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

echo "Enter a limit"

read limit

echo "prime numbers upto $limit are :"

echo "1"

i=2

while [ $i -le $limit ]

do

flag=1

j=2

while [ $j -lt $i ]

do

rem=$(( $i % $j ))

if [ $rem -eq 0 ]

then

flag=0

break

fi

j=$(( $j+1 ))

done

if [ $flag -eq 1 ]

then

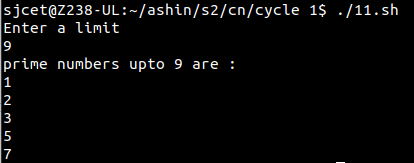
echo "$i"

fi

i=$(( $i+1 ))

done

Output



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

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))

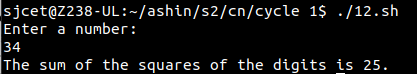
number=$((number / 10))

done

# Output the result

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

Output



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

clear

echo "Entre a string to find the number of Vowels "

read st

len=`expr $st | wc -c`

len=`expr $len - 1`

count=0

while [ $len -gt 0 ]

do

ch=`expr $st | cut -c $len`

case $ch in

[aeiou,AEIOU]) count=`expr $count + 1` ;;

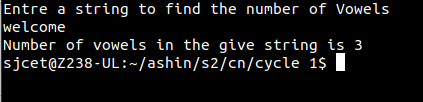
esac

len=`expr $len - 1`

done

echo "Number of vowels in the give string is $count"

Output



1. Write a Shell program to display student grades.

clear

echo -----------------------------------

echo '\tStudent Mark List'

echo -----------------------------------

echo Enter the Student name

read name

echo Enter the Register number

read rno

echo Enter the Mark1

read m1

echo Enter the Mark2

read m2

echo Enter the Mark3

read m3

echo Enter the Mark4

read m4

echo Enter the Mark5

read m5

tot=$(expr $m1 + $m2 + $m3 + $m4 + $m5)

avg=$(expr $tot / 5)

echo -----------------------------------

echo '/t Student Mark List'

echo -----------------------------------

echo "Student Name : $name"

echo "Register Number : $rno"

echo "Mark1 : $m1"

echo "Mark2 : $m2"

echo "Mark3 : $m3"

echo "Mark4 : $m4"

echo "Mark5 : $m5"

echo "Total : $tot"

echo "Average : $avg"

if [ $m1 -ge 35 ] && [ $m2 -ge 35 ] && [ $m3 -ge 35 ] && [ $m4 -ge 35 ] && [ $m5 -ge 35 ]

then

echo "Result : Pass"

if [ $avg -ge 90 ]

then

echo "Grade : S"

elif [ $avg -ge 80 ]

then

echo "Grade : A"

elif [ $avg -ge 70 ]

then

echo "Grade : B"

elif [ $avg -ge 60 ]

then

echo "Grade : C"

elif [ $avg -ge 50 ]

then

echo "Grade : D"

elif [ $avg -ge 35 ]

then

echo "Grade : E"

fi

else

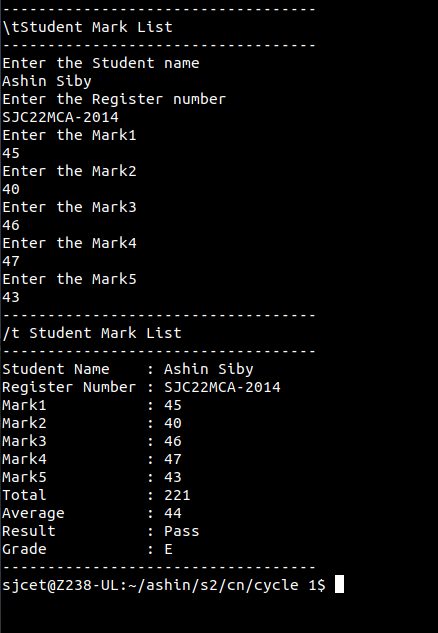
echo "Result : Fail"

fi

echo -----------------------------------

59,29 Bot

Output



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

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

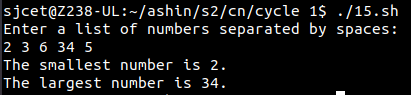
done

# Output the result

echo "The smallest number is $min."

echo "The largest number is $max."

Output



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

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

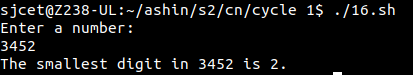
fi

done

# Output the result

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

Output



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.

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."

Output



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

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

read numbers

# Convert the space-separated string to an array

arr=($numbers)

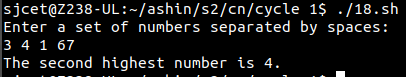
# Sort the array in descending order

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

# Output the second highest number

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

Output



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

# 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

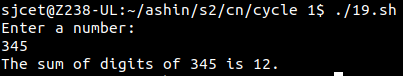
# Call the function to calculate the sum of digits

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

# Output the result

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

Output



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

# 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

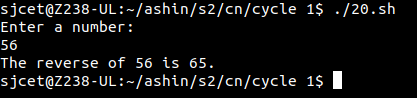
# Call the function to reverse the number

result=$(reverse\_number $num)

# Output the result

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

Output



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

# 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

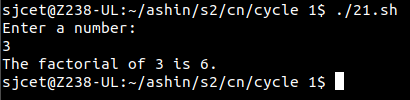
factorial=$((factorial \* i))

done

# Output the result

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

Output



1. Write a Shell program to generate Fibonacci series.

# 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

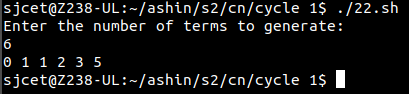
a=$b

b=$c

done

echo

Output



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.

if [ $# -ne 2 ]; then

echo "Usage: $0 file1 file2"

exit 1

fi

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

rm "$2"

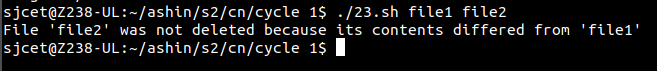
echo "File '$2' deleted because its contents were identical to '$1'"

else

echo "File '$2' was not deleted because its contents differed from '$1'"

fi

Output



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

#!/bin/bash

while true; do

clear

echo "======================="

echo " MAIN MENU "

echo "======================="

echo "1. List current directory"

echo "2. Print working directory"

echo "3. Display date"

echo "4. Display users logged in"

echo "5. Exit"

echo -n "Enter your choice: "

read choice

case $choice in

1)

ls -la

echo "Press enter to continue"

read

;;

2)

pwd

echo "Press enter to continue"

read

;;

3)

date

echo "Press enter to continue"

read

;;

4)

who

echo "Press enter to continue"

read

;;

5)

echo "Exiting..."

exit 0

;;

\*)

echo "Invalid choice. Press enter to continue"

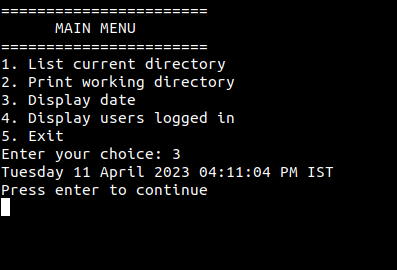
read

;;

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.

#!/bin/bash

# Loop through all files in the current directory

for file in \*; do

# Check if the file is executable

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

# If the file is not executable, make it executable

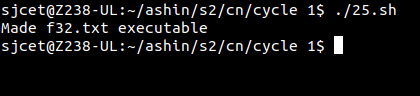
chmod +x "$file"

echo "Made $file executable"

fi

done

Output



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

#!/bin/bash

for i in 1 2 3; do

for j in 1 2 3; do

for k in 1 2 3; do

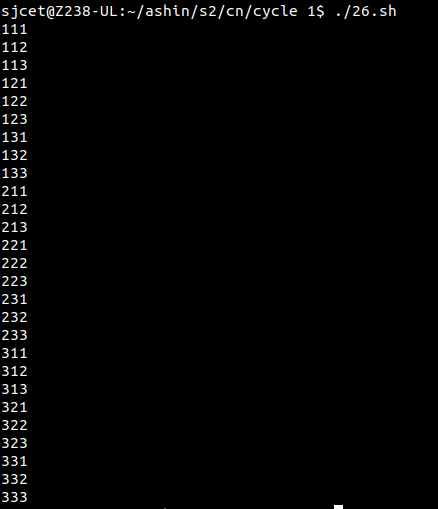
echo "$i$j$k"

done

done

done

Output



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

1

2 3

4 5 6

7 8 9 10

#!/bin/bash

rows=4

current=1

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

do

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

do

echo -n "$current "

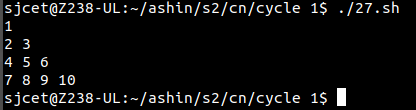
(( current++ ))

done

echo

done

Output



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

#!/bin/bash

# Function to calculate the binomial coefficient

function binom {

if [ $2 -eq 0 ] || [ $2 -eq $1 ]; then

echo 1

else

echo $(( $(binom $(($1-1)) $(($2-1))) + $(binom $(($1-1)) $2) ))

fi

}

# Get the number of rows from the user

echo "Enter the number of rows in Pascal's triangle: "

read rows

# Loop through each row

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

# Loop through each element in the row

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

# Calculate the binomial coefficient and print

val=$(binom $i $j)

echo -n "$val "

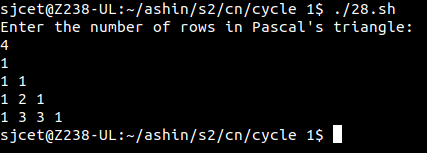
done

# Move to next row

echo ""

done

Output



1. Write a Decimal to Binary Conversion Shell Script

#!/bin/bash

# Prompt user for decimal input

read -p "Enter decimal number: " decimal

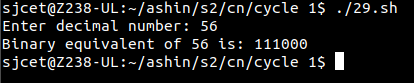
# Convert decimal to binary using 'bc' command

binary=$(echo "obase=2;$decimal" | bc)

# Print binary result

echo "Binary equivalent of $decimal is: $binary"

Output



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

#!/bin/bash

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

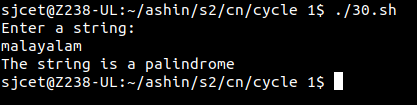
echo "The string is a palindrome"

else

echo "The string 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.

#!/bin/bash

filename="$1"

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

echo "File not found!"

exit 1

fi

# Use tr to convert everything to lowercase and remove punctuation

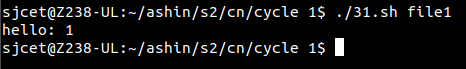
# Use awk to print each word on a separate line

# Use sort and uniq to get unique words and count their occurrences

# Use awk to format the output as "word: count"

cat "$filename" | tr '[:upper:]' '[:lower:]' | tr -s '[:punct:]' ' ' | awk '{for(i=1;i<=NF;i++){print $i}}' | sort | uniq -c | awk '{print $2 ": " $1}'

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.

#!/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)

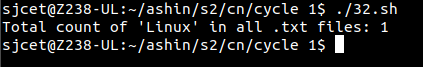
count=$((count + occurrences))

done

# Print the total count

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

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.

#!/bin/bash

read -p "Enter your password: " password

# Check if password is at least 8 characters long

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

echo "Password length must be at least 8 characters."

exit 1

fi

# Check if password contains both alphabet and number

if ! [[ "$password" =~ [A-Za-z]+[0-9]+ ]]; then

echo "Password must contain both alphabet and number."

exit 1

fi

# Check if password includes both small and capital case letters

if ! [[ "$password" =~ [a-z]+ ]] || ! [[ "$password" =~ [A-Z]+ ]]; then

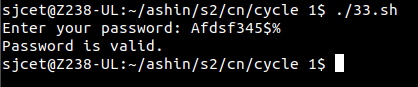
echo "Password must include both small and capital case letters."

exit 1

fi

echo "Password is valid."

Output



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

echo "Enter directory path: "

read directory

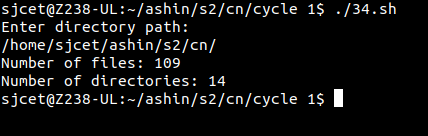
num\_files=$(find $directory -type f | wc -l)

num\_directories=$(find $directory -type d | wc -l)

echo "Number of files: $num\_files"

echo "Number of directories: $num\_directories"

Output



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

#!/bin/bash

# Define a list of strings

my\_list=("string1" "string2" "string3" "string4")

# Reverse the order of the list

my\_list=($(echo "${my\_list[@]}" | tr ' ' '\n' | tac | tr '\n' ' '))

# Reverse each string in the list

for i in "${!my\_list[@]}"

do

my\_list[$i]=`echo ${my\_list[$i]} | rev`

done

# Print the reversed list of strings

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

Output

