Q.1 Write a shell script that does the following: 1.Use a for loop to print the numbers from 1 to 5. 2.After the loop ends, print a message saying "Loop completed."

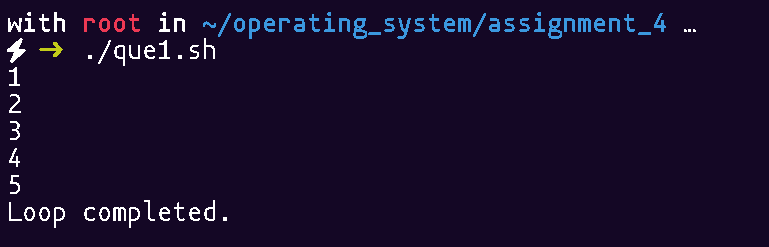
for i in $(seq 1 5); do

echo $i

done

echo "Loop completed."

OUTPUT:



Q.2 Write a shell script that does the following: 1.Use a for loop to iterate over the numbers from 1 to 10. 2.For each number: Print whether the number is even or odd.

for i in $(seq 1 10); do

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

echo "$i is even"

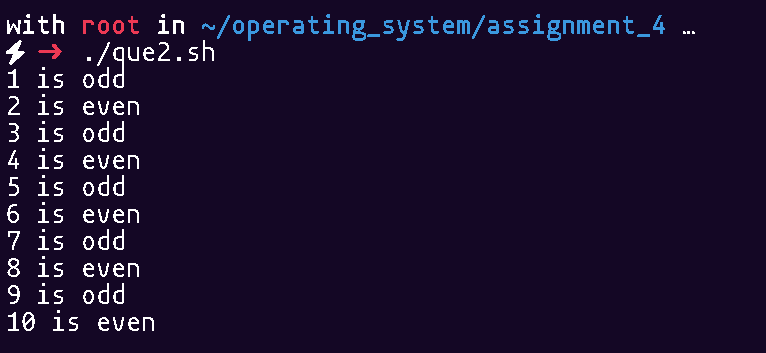
else

echo "$i is odd"

fi

done

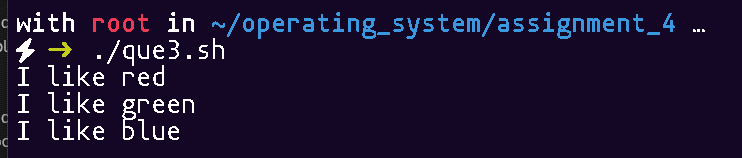
OUTPUT:



Q.3 Write a shell script that does the following: Use a for loop to loop through a list of three colors: "red", "green", and "blue". For each color, print a message saying "I like [color]".

for color in red green blue; do echo "I like $color"; done

OUTPUT:



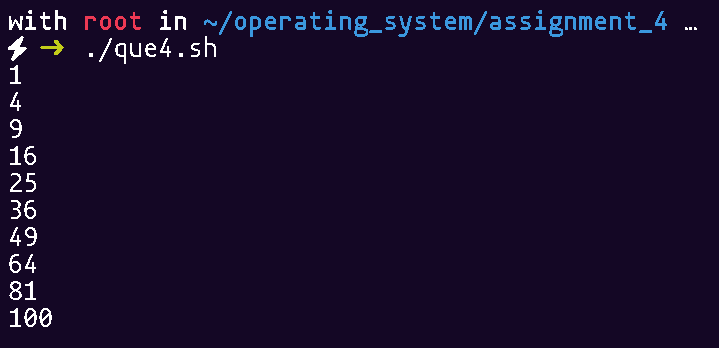
Q.4 Write a shell script that does the following: 1. Use a for loop to iterate over the numbers from 1 to 10. 2. Inside the loop, calculate the square of each number and print the result.

for i in $(seq 1 10); do

echo $((i \* i))

done

OUTPUT:



Q.5 Write a shell script that does the following: Use a for loop to iterate over the numbers from 5 to 15. For each number: o Check if the number is divisible by 3. o If it is divisible by 3, print "Number [x] is divisible by 3". o If it is not divisible by 3, print "Number [x] is not divisible by 3".

for i in $(seq 5 15); do

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

echo "Number $i is divisible by 3"

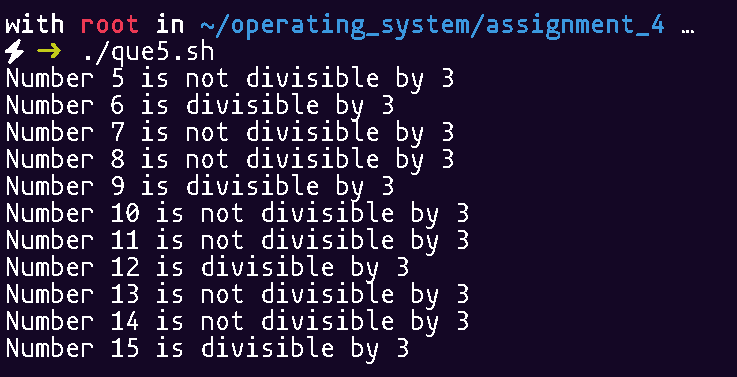
else

echo "Number $i is not divisible by 3"

fi

done

OUTPUT:



Q.6 Write a shell script that does the following: 1. Use a for loop to iterate through the numbers from 1 to 20. 2. For each number: o Check if the number is prime. o If the number is prime, print "Number [x] is prime." o If the number is not prime, print "Number [x] is not prime."

for num in $(seq 1 20); do

count=0

for i in $(seq 1 $num); do

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

count=$((count + 1))

fi

done

if [ $count -eq 2 ]; then

echo "Number $num is prime"

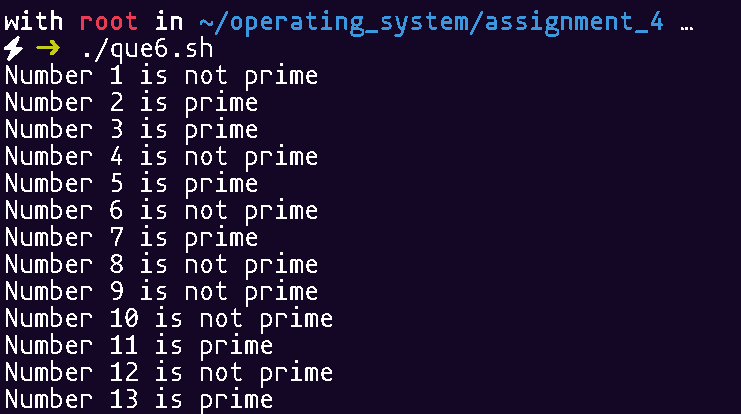
else

echo "Number $num is not prime"

fi

done

OUTPUT:



Q.7 Write a Shell Script to Reverse a Number

read n

rev=0

while [ $n -gt 0 ]; do

r=$((n % 10))

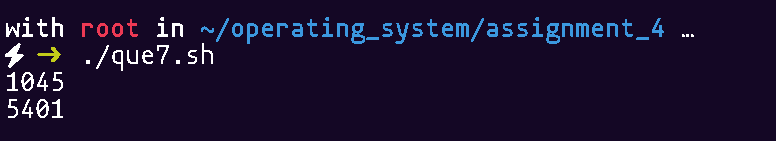
rev=$((rev \* 10 + r))

n=$((n / 10))

done

echo $rev

OUTPUT:



Q.8 Write a Shell Script to Find the Length of a String Create a shell script that prompts the user to enter a string of your name and then calculates and displays its length using a while loop

read str

len=0

while [ "$str" != "${str%?}" ]; do

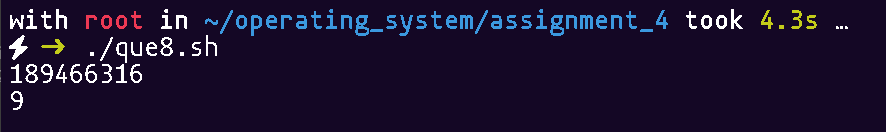
str=${str%?}

len=$((len + 1))

done

echo $len

OUTPUT:



Q.9 Write a Shell Script to Print Even Numbers Create a shell script that does the following: 1. Print your name and roll no. 2. Initializes a counter variable to 0. Uses a while loop to print all even numbers from 0 to 20. Increment the counter by 2 in each iteration of the loop.

echo "Name: Shiv Arora"

echo "Roll No: 4526"

i=0

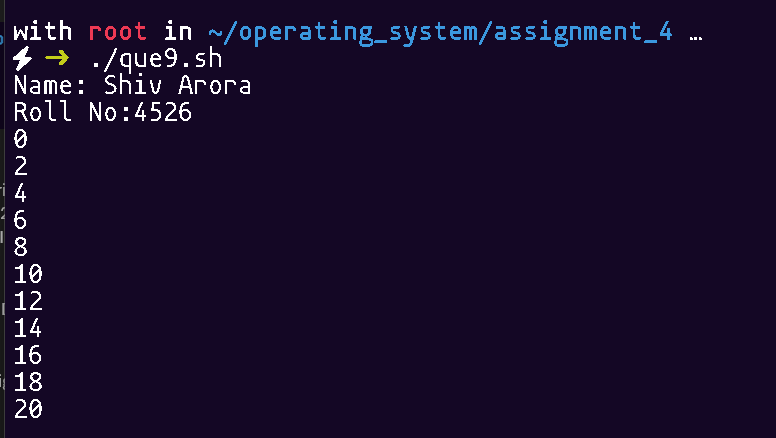
while [ $i -le 20 ]; do

echo $i

i=$((i + 2))

done

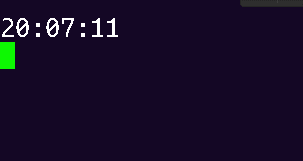
OUTPUT:



Q.10 Case Study1: Create a Digital Clock using Shell Script.

while true; do clear; date +"%T"; sleep 1; done

OUTPUT:



Q.11 Case Study2: Create Sign Up Form in Shell Scripting

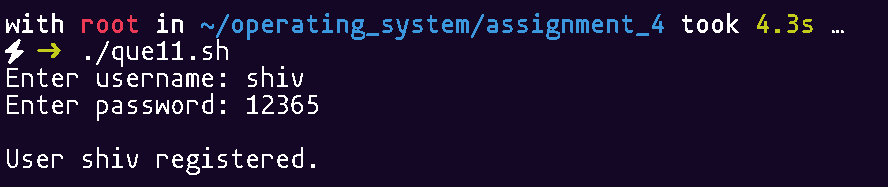
read -p "Enter username: " user

read -sp "Enter password: " pass

echo

echo "User $user registered."

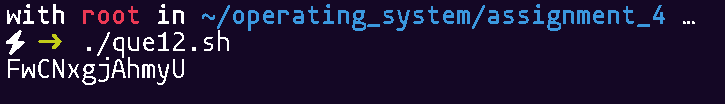
OUTPUT:



Q.12 Case Study 3 :Create a Password Generator using Shell Script.

< /dev/urandom tr -dc A-Za-z0-9 | head -c 12; echo

OUTPUT:



Q.13 Case-Study 4: Create a Student Database perform following operations: 1. Create a Database. 2. View a Database. 3. Insert a Student Record. 4. Update a Record. 5. Delete a Record. 6. Show Record Part

DB="students.txt"

create\_db() {

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

touch "$DB"

echo "Database created."

else

echo "Database already exists."

fi

}

view\_db() {

if [ -f "$DB" ]; then

echo "----- Student Database -----"

cat "$DB"

else

echo "Database not found."

fi

}

insert\_record() {

echo "Enter Roll Number:"

read roll

echo "Enter Name:"

read name

echo "Enter Branch:"

read branch

echo "$roll,$name,$branch" >> "$DB"

echo "Record inserted."

}

update\_record() {

echo "Enter Roll Number to update:"

read roll

if grep -q "^$roll," "$DB"; then

echo "Enter New Name:"

read name

echo "Enter New Branch:"

read branch

sed -i "/^$roll,/c\\$roll,$name,$branch" "$DB"

echo "Record updated."

else

echo "Record not found."

fi

}

delete\_record() {

echo "Enter Roll Number to delete:"

read roll

if grep -q "^$roll," "$DB"; then

sed -i "/^$roll,/d" "$DB"

echo "Record deleted."

else

echo "Record not found."

fi

}

search\_record() {

echo "Enter Roll Number to search:"

read roll

grep "^$roll," "$DB" || echo "Record not found."

}

while true; do

echo "

------ Student Database Menu ------

1. Create Database

2. View Database

3. Insert Student Record

4. Update Record

5. Delete Record

6. Show Specific Record

7. Exit

-----------------------------------"

echo "Enter choice:"

read choice

case $choice in

1) create\_db ;;

2) view\_db ;;

3) insert\_record ;;

4) update\_record ;;

5) delete\_record ;;

6) search\_record ;;

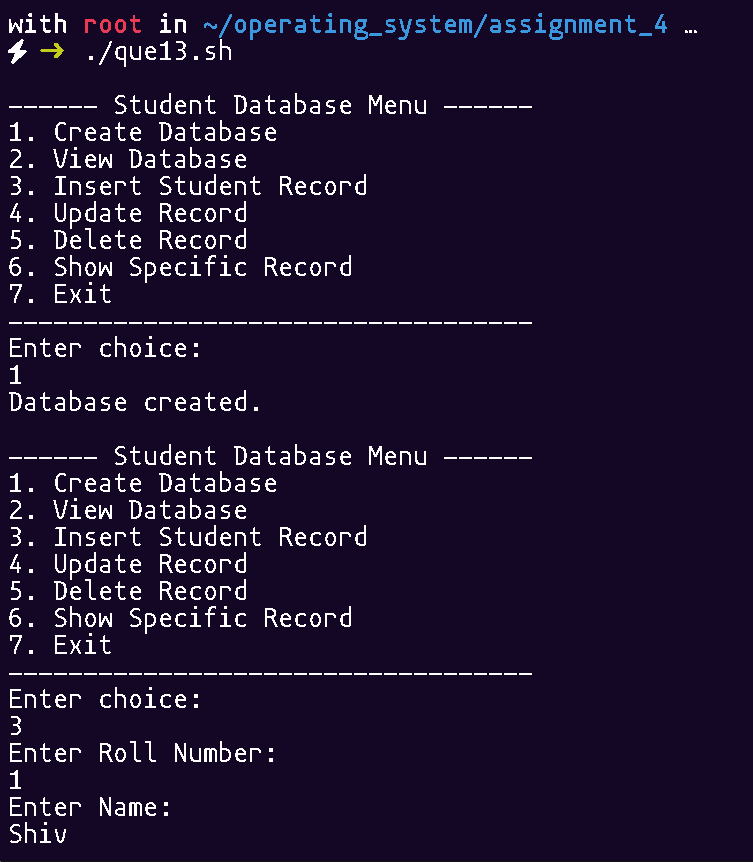
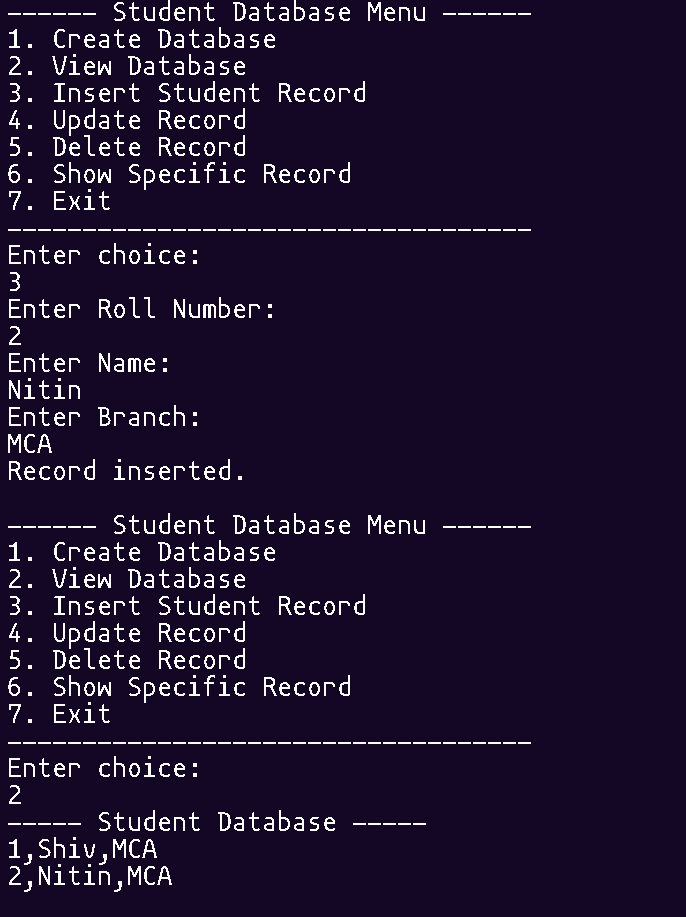
7) echo "Exiting..."; break ;;

\*) echo "Invalid choice." ;;

esac

done

OUTPUT:

Q.14 Write a shell script that receives any number of file names as arguments checks if every argument supplied is a file or a directory and reports accordingly. whenever the argument is a file or directory.

for item in "$@"; do

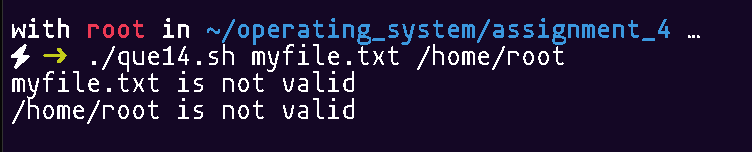
if [ -f "$item" ]; then echo "$item is a file"

elif [ -d "$item" ]; then echo "$item is a directory"

else echo "$item is not valid"; fi

done

OUTPUT:

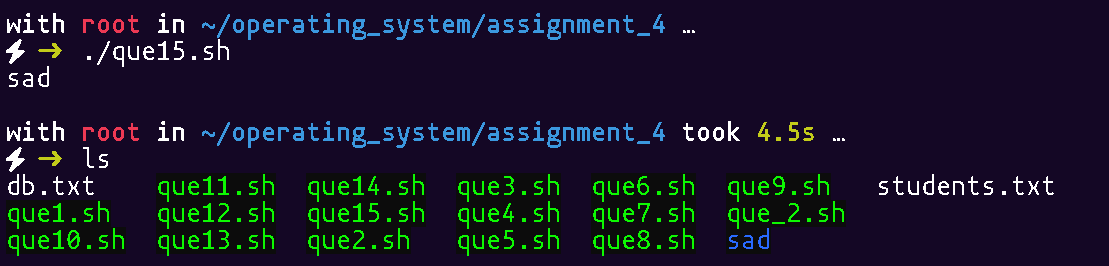


Q.15 Write Shell Script Make directory by checking existence:

read dir

[ ! -d "$dir" ] && mkdir "$dir"

OUTPUT:



Q.16 Write Shell Script for Add array elements - Shell Script

set -- 1 2 3 4 5

sum=0

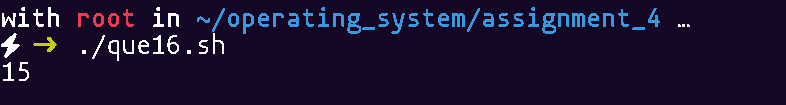
for i; do

sum=$((sum + i))

done

echo "$sum"

OUTPUT:



Q.17 Write Shell Script reverse elements in array

#!/bin/bash

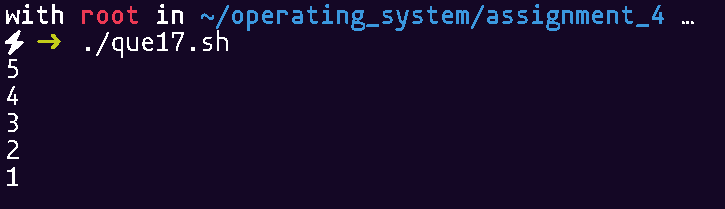
arr=(1 2 3 4 5)

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

echo "${arr[i]}"

done

OUTPUT:



Q.18 Write shell script program to addition, subtraction, multiplication of two matrix.

#!/bin/bash

# Function to read a matrix

read\_matrix() {

local name=$1

local -n matrix=$name

echo "Enter elements for Matrix $name:"

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

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

read -p "$name[$i][$j] = " matrix[$i,$j]

done

done

}

# Function to print a matrix

print\_matrix() {

local name=$1

local -n matrix=$name

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

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

echo -n "${matrix[$i,$j]} "

done

echo

done

}

# Function for addition

add\_matrices() {

echo "Addition of matrices:"

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

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

echo -n "$((a[$i,$j] + b[$i,$j])) "

done

echo

done

}

# Function for subtraction

subtract\_matrices() {

echo "Subtraction of matrices (A - B):"

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

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

echo -n "$((a[$i,$j] - b[$i,$j])) "

done

echo

done

}

# Function for multiplication

multiply\_matrices() {

echo "Multiplication of matrices:"

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

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

sum=0

for ((k=0; k<cols; k++)); do

sum=$((sum + a[$i,$k] \* b[$k,$j]))

done

echo -n "$sum "

done

echo

done

}

# Main program

declare -A a

declare -A b

read -p "Enter number of rows: " rows

read -p "Enter number of columns: " cols

read\_matrix a

read\_matrix b

echo

add\_matrices

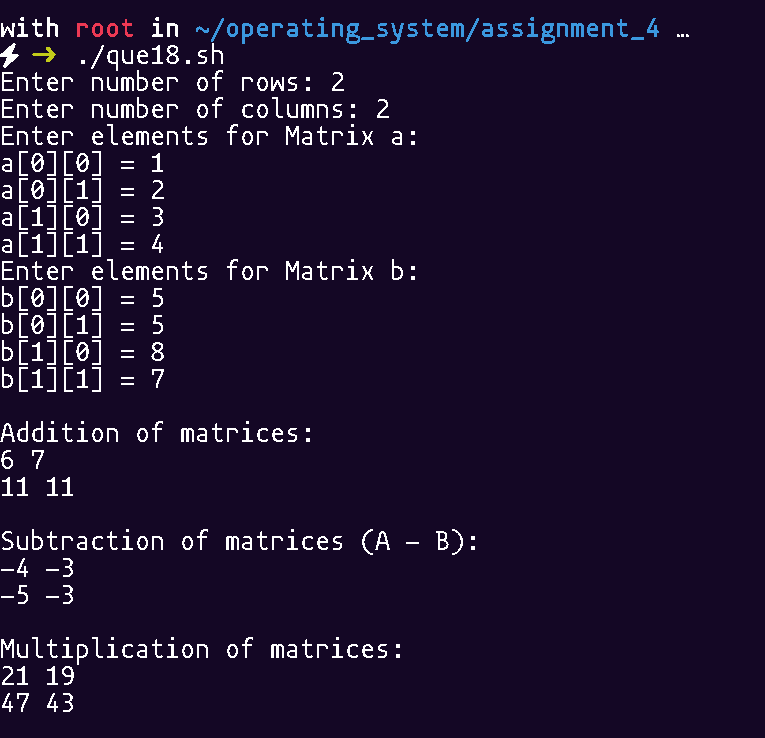
echo

subtract\_matrices

echo

multiply\_matrices

OUTPUT:



Q.19 Case Study : User Management Script Create a script to manage users — add, delete, and list users using Function

#!/bin/bash

# Check if running as root

if [[ $EUID -ne 0 ]]; then

echo "This script must be run as root. Use sudo."

exit 1

fi

# Function to add a user

add\_user() {

read -p "Enter username to add: " username

if id "$username" &>/dev/null; then

echo "User '$username' already exists."

else

useradd "$username"

if [[ $? -eq 0 ]]; then

echo "User '$username' added successfully."

else

echo "Failed to add user."

fi

fi

}

# Function to delete a user

delete\_user() {

read -p "Enter username to delete: " username

if id "$username" &>/dev/null; then

userdel "$username"

if [[ $? -eq 0 ]]; then

echo "✅ User '$username' deleted successfully."

else

echo "Failed to delete user."

fi

else

echo "User '$username' does not exist."

fi

}

# Function to list all users

list\_users() {

echo " Listing all users:"

cut -d: -f1 /etc/passwd

}

# Function to display menu

show\_menu() {

echo "========= User Management ========="

echo "1. Add User"

echo "2. Delete User"

echo "3. List Users"

echo "4. Exit"

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

}

# Main loop

while true; do

show\_menu

read -p "Enter your choice [1-4]: " choice

case $choice in

1) add\_user ;;

2) delete\_user ;;

3) list\_users ;;

4) echo "Exiting..."; exit 0 ;;

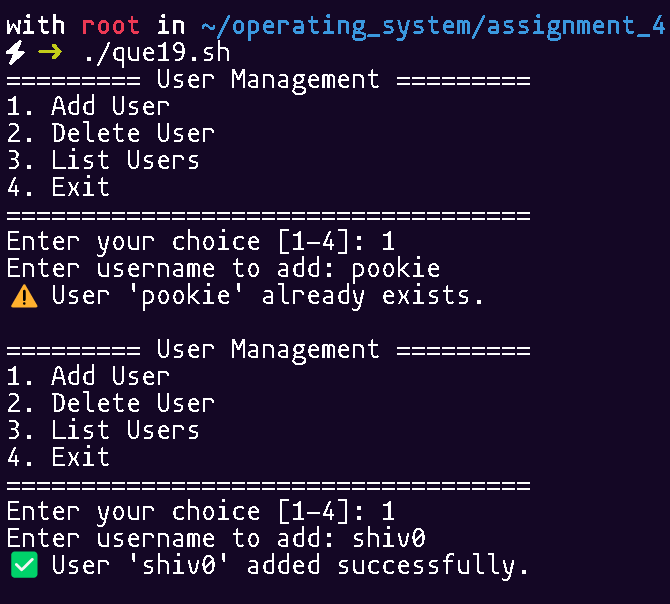
\*) echo "Invalid choice. Try again." ;;

esac

echo

done

OUTPUT:



Q.20 Case Study : System Health Check Script Create a Shell Script to Monitor system health — CPU load, memory usage, disk space using Function.

cpu=$(top -bn1 | grep "Cpu(s)" | awk '{print $2+$4}')

mem=$(free -m | awk '/Mem:/ {printf "%d/%dMB (%.2f%%)", $3, $2, $3\*100/$2 }')

disk=$(df -h / | awk 'NR==2 {print $5}')

echo "CPU: $cpu%"; echo "Memory: $mem"; echo "Disk: $disk"

OUTPUT:

