**# 1. Generate Triangle Output**

# Write a shell script to generate the following output:

# \*

# \*\*

# \*\*\*

# \*\*\*\*

#!/bin/bash

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

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

echo -n "\*"

done

echo

done

**# 2. Greeting Message**

# Accept the first name, middle name, and last name of a person in variables fname, mname, and lname. Greet the person using an appropriate message.

#!/bin/bash

echo "Enter first name:"

read fname

echo "Enter middle name:"

read mname

echo "Enter last name:"

read lname

echo "Hello, $fname $mname $lname! Welcome!"

**# 3. File Sizes in Current Directory**

# Display the name of files in the current directory along with the names of files with maximum and minimum size.

#!/bin/bash

ls -l | awk '{print $9, $5}' | tail -n +2 | sort -k2 -n

largest=$(ls -S | head -n 1)

smallest=$(ls -Sr | head -n 1)

echo "Largest file: $largest"

echo "Smallest file: $smallest"

**# 4. Check Working Day**

# Write a script that checks whether it is a working day (Monday-Friday).

#!/bin/bash

day=$(date +%u)

if ((day >= 1 && day <= 5)); then

echo "It's a working day."

else

echo "It's not a working day."

fi

**# 5. Health Club Membership**

# Write a script that accepts a member into MBT health club if their weight is within 30-250 kg.

#!/bin/bash

echo "Enter your weight (kg):"

read weight

if ((weight >= 30 && weight <= 250)); then

echo "Welcome to MBT Health Club!"

else

echo "Sorry, your weight is not within the acceptable range."

fi

**# 6. Greet User Based on Time**

# Write a shell script to greet the user depending on the system time.

#!/bin/bash

hour=$(date +%H)

if ((hour < 12)); then

echo "Good Morning!"

elif ((hour < 18)); then

echo "Good Afternoon!"

else

echo "Good Evening!"

fi

**# 7. Modify Student Records**

# Write a script that accepts a roll number, searches for it in a data file, and allows modifying name and marks if found.

#!/bin/bash

echo "Enter roll number:"

read rollno

file="file"

if grep -q "^$rollno:" $file; then

echo "Record found. Enter new name and marks for 3 subjects:"

read name marks1 marks2 marks3

sed -i "/^$rollno:/c\\$rollno:$name:$marks1:$marks2:$marks3" $file

echo "Record updated."

else

echo "Roll No Not Found"

fi

**# 8. Accept RollNo from Command Line**

# Modify the previous program to accept RollNo from the command line.

#!/bin/bash

rollno=$1

file="file"

if grep -q "^$rollno:" $file; then

echo "Record found. Enter new name and marks for 3 subjects:"

read name marks1 marks2 marks3

sed -i "/^$rollno:/c\\$rollno:$name:$marks1:$marks2:$marks3" $file

echo "Record updated."

else

echo "Roll No Not Found"

fi

**# 9. Delete Record**

# Modify the program to delete a record after confirmation.

#!/bin/bash

echo "Enter roll number:"

read rollno

file="file"

if grep -q "^$rollno:" $file; then

grep "^$rollno:" $file

echo "Do you want to delete this record? (yes/no)"

read confirm

if [[ $confirm == "yes" ]]; then

sed -i "/^$rollno:/d" $file

echo "Record deleted."

else

echo "Deletion cancelled."

fi

else

echo "Roll No Not Found"

fi

**# 10. File Type Script**

# Write a script that takes a command line argument and reports on its file type.

#!/bin/bash

if [[ $# -ne 1 ]]; then

echo "Error: Provide exactly one argument."

exit 1

fi

if [[ -f $1 ]]; then

echo "$1 is a regular file."

elif [[ -d $1 ]]; then

echo "$1 is a directory."

else

echo "$1 is of other type."

fi

**# 11. Validate Student Records**

# Write a script to validate student records.

#!/bin/bash

file="student"

log="log1"

while read -r line; do

roll=$(echo "$line" | cut -d':' -f1)

marks=$(echo "$line" | cut -d':' -f3-5)

if grep -q "^$roll:" $file; then

echo "$line - roll number exists" >>$log

elif ! [[ $marks =~ ^([1-9][0-9]?|99)$ ]]; then

echo "$line - marks out of range" >>$log

else

total=$(echo "$marks" | awk -F':' '{print $1+$2+$3}')

percentage=$((total / 3))

echo "$line - Total: $total, Percentage: $percentage%"

fi

done <"$file"

**# 12. Update Batch Code**

# Write a script to update the batch code in the master file.

#!/bin/bash

master="master"

echo "Enter batch code:"

read batch

if grep -q "^$batch:" $master; then

echo "Enter new records (RollNo:Name:Marks\_Hindi:Marks\_Maths:Marks\_Physics):"

read -p "Record: " record

file="$batch"

echo "$record" >>"$file"

count=$(wc -l <"$file")

sed -i "/^$batch:/s/[0-9]\*$/$count/" $master

echo "Record added and master file updated."

else

echo "Batch code not found."

fi

**# 13. Library Functions**

# Create a library of shell functions.

concat\_strings() {

echo "$1$2"

}

string\_length() {

echo "${#1}"

}

compare\_strings() {

[[ $1 == $2 ]] && echo "Strings are equal" || echo "Strings are not equal"

}

is\_palindrome() {

[[ $1 == $(echo "$1" | rev) ]] && echo "Palindrome" || echo "Not a palindrome"

}

reverse\_string() {

echo "$1" | rev

}

**# 14. Assignments on `sed`**

# Perform the following operations on EmpData.

a. Replace first 3 lines' delimiters:

sed '1,3s/|/:/g' EmpData

b. Replace all delimiters:

sed 's/|/:/g' EmpData

c. Insert string in first line:

sed '1s/^/TechM Employees\n/' EmpData

d. Store directors, d.g.m, g.m in separate files:

grep 'director' EmpData >directors

grep 'd.g.m' EmpData >dgms

grep 'g.m.' EmpData >gms

e. Store first 4 lines:

sed -n '1,4p' EmpData >Empupdate

f. Replace "account" with "accounts":

sed 's/account/accounts/g' EmpData

g. Select lines not having "g.m":

grep -v 'g.m' EmpData

h. Insert blank line after every line:

sed G EmpData