Name – Vaishnavi Kuldharme

1 Write a shell script which will generate the O/P as follows

```
*
**
***
****
```

```
nano star_pattern
#!/bin/bash

# Number of rows

Lines=4

# Loop to print the pattern

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

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

echo -n "*"

done

echo ""

done
chmod +x star
./star
```

2 Accept the first name, middle name, and last name of a person in variables fname, mname and lname respectively. Greet the person (take his full name) using appropriate message.

```
#!/bin/bash


# Accept first last middle name


read -p "Enter The First Name: " First_name
```

```
read -p "Enter The Middle Name: " Middle_name

read -p "Enter The Last Name: " Last_name

# Display

echo "Hello, $fname $mname $lname! Welcome to Shell Scripting."
```

```
[admin@hostname01 Desktop]$ vim shreya.sh
[admin@hostname01 Desktop]$ chmod +x shreya.sh
[admin@hostname01 Desktop]$ ./shreya.sh
Enter First Name: shreya
Enter Middle Name: singh
Enter Last Name: singh
Hello, shreya singh singh! Welcome to Shell Scripting.
```

3 Display the name of files in the current directory along with the names of files with maximum & minimum size. The file size is considered in bytes.

```
#!/bin/bash



# Display all files with their sizes

echo "Files in the Current Directory:"

ls -lS --block-size=1 | awk '{print $5, $9}' | tail -n +2



# Get the smallest and largest files

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

min_file=$(ls -Sr | head -n 1)
```

```
# Get their sizes in bytes

max_size=$(stat -c%s "$max_file")

min_size=$(stat -c%s "$min_file")


# Display Results

echo -e "\n Largest File: $max_file ($max_size bytes)"

echo " Smallest File: $min_file ($min_size bytes)"
```

```
[admin@hostname01 Desktop]$ vim filesize.sh
[admin@hostname01 Desktop]$ chmod +x filesize.sh
[admin@hostname01 Desktop]$ ./filesize.sh
Files in the Current Directory:
7744 lsdoc
479 filesize.sh
315 users
262 shreya.sh
215 friends
215 newfriends
111 triangle.sh
20 data.txt
20 demo

 Largest File: lsdoc (7744 bytes)
 Smallest File: demo (20 bytes)
```

4 Write a script which when executed checks out whether it is a working day or not?

(Note: Working day Mon-Fri)

```
#!/bin/bash
day =$(date +%u)
if [$day -ge 1] && [$day -le 5]; then
        echo "Weekday"
else
        echo "Weekend"
fi
```

5 Write a script that accepts a member into HP health club, if the weight of the person is withing the range of 30-250 Kgs.

```
#!/bin/bash

echo -n "Enter your weight in Kgs: "

read weight

if [[ $weight -ge 30 && $weight -le 250 ]]; then

echo "Congratulations! We Welcome you to HP HEALTH CLUB."

else

echo "Sorry, your weight doesnt fits the category. "

fi
[admin@hostname01 Desktop]$ vim health.sh
[admin@hostname01 Desktop]$ chmod +x health.sh
[admin@hostname01 Desktop]$ ./health.sh
Enter your weight in Kgs: 50
Congratulations! We Welcome you to HP HEALTH CLUB.
```

6 Write a shell script that greets the user with an appropriate message depending on the system time.

```
[admin@hostname01 Desktop]$ vim greeting.sh
[admin@hostname01 Desktop]$ chmod +x greeting.sh
```

```
[admin@hostname01 Desktop]$ ./greeting.sh
Good evening
```

```
#!/bin/bash

hour=$(date +%H)

if [ $hour -ge 5 ] && [ $hour -lt 12 ]; then
    echo "Good morning"
elif [ $hour -ge 12 ] && [ $hour -lt 17 ]; then
    echo "Good afternoon"
elif [ $hour -ge 17 ] && [ $hour -lt 21 ]; then
    echo "Good evening"
else
    echo "Good night"
fi
```

7 A data file file has some student records including rollno, names and subject marks. The fields are separated by a ":". Write a shell script that accepts roll number from the user, searches it in the file and if the roll number is present - allows the user to modify name and marks in 3 subjects.
If the roll number is not present, display a message "Roll No Not Found". Allow the user to modify one record at a time.

```
#!/bin/bash

daya_file="students.txt"

if [[ ! -f $data_file ]]; then
    echo "Data file not found"
    exit 1
fi

echo -n "Enter roll number: "
read roll_no

record=$(grep "^$roll_no:" $data_file)
```

```
if [[ -z $record ]]; then
    echo "Roll number $roll_no not found"
    exit 1

else
    echo "Record found: $record"
    current_name=$(echo $record | cut -d':' -f2)
    current_marks=$(echo $record | cut -d':' -f3)
    echo -n "Enter new name"
    read new_name

    echo -n "Enter marks for subject 1"
    read marks1

    echo -n "Enter marks for subject 2"
    read marks2

    echo -n "Enter marks for subject 3"
    read marks3

    new_record="$roll_no:$new_name:$marks1:$marks2:$marks3"

    sed -i "s/^$record\$/$new_record/" $data_file

    echo "Record updated"
fi
```

8 Modify program 7 to accept the RollNo from the command line.

```
#!/bin/bash

# File containing student records

data_file="students.txt"
```

```
# Check if the roll number is provided as a command-line argument

if [ -z "$1" ]; then

echo "Usage: $0 <rollno>"

exit 1

fi

# Get the roll number from the command-line argument

rollno=$1

# Search for the roll number in the file using grep

record=$(grep "^$rollno:" "$data_file")

if [ -z "$record" ]; then

echo "Roll No Not Found"

else

# Display the current record

echo "Current record: $record"

# Prompt the user to enter the new name and marks

echo "Enter the new name:"
```

```
read new_name

echo "Enter the new marks for subject 1:"

read new_marks1

echo "Enter the new marks for subject 2:"

read new_marks2

echo "Enter the new marks for subject 3:"

read new_marks3

# Create the new record

new_record="$rollno:$new_name:$new_marks1:$new_marks2:$new_marks3"

# Replace the old record with the new record in the file using sed

sed -i "s/^$rollno:.*/$new_record/" "$data_file"

echo "Record updated successfully."

fi
```

9 Modify the program 7 to accept the RollNo and display the record and ask for delete confirmation. Once confirmed delete the record and update the data file.

```
#!/bin/bash

# File containing student records

data_file="students.txt"
```

```
# Check if the roll number is provided as a command-line argument

if [ -z "$1" ]; then

echo "Usage: $0 <rollno>"

exit 1

fi

# Get the roll number from the command-line argument

rollno=$1

# Search for the roll number in the file using grep

record=$(grep "^$rollno:" "$data_file")

if [ -z "$record" ]; then

echo "Roll No Not Found"

else

# Display the current record

echo "Current record: $record"

# Ask for delete confirmation

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

read confirmation
```

```
if [ "$confirmation" = "yes" ]; then

# Delete the record from the file using sed

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

echo "Record deleted successfully."

else

echo "Deletion cancelled."

fi

fi
```

10 Write a script that takes a command line argument and reports on its file type (regular file, directory file, etc.). For more than one argument generate error message.

```
#!/bin/bash

# Check if more than one argument is provided

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

echo "Usage: $0 <filename>"

exit 1

fi

# Get the filename from the command-line argument

filename=$1
```

```
# Check if the file exists

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

echo "File does not exist."

exit 1

fi

# Determine the file type

if [ -f "$filename" ]; then

echo "$filename is a regular file."

elif [ -d "$filename" ]; then

echo "$filename is a directory."

elif [ -L "$filename" ]; then

echo "$filename is a symbolic link."

else

echo "$filename is of another file type."

fi
```

11 Add some student records in the “student” file manually. The fields to be considered are “RollNo”, “Name”, “Marks_Hindi”, “Marks_Maths”, “Marks_Physics”.
Write a script which does the following

a If the roll number already exists, then store the record and the following message “roll number exists” in a log file “log1”.

b If the marks in the subjects is not in the range of 1 – 99 then store such a record followed by a message “marks out of range” in “log1”

c If the data is valid, the calculate total, percentage, grade and display on the terminal

```
#!/bin/bash

# File containing student records

data_file="student"

log_file="log1"

# Function to calculate grade based on percentage

calculate_grade() {

local percentage=$1

if (( $(echo "$percentage >= 90" | bc -l) )); then

echo "A"

elif (( $(echo "$percentage >= 80" | bc -l) )); then

echo "B"

elif (( $(echo "$percentage >= 70" | bc -l) )); then

echo "C"

elif (( $(echo "$percentage >= 60" | bc -l) )); then

echo "D"

else

echo "F"

fi

}
```

```
# Prompt the user to enter student details

echo "Enter Roll Number:"

read rollno

echo "Enter Name:"

read name

echo "Enter Marks in Hindi:"

read marks_hindi

echo "Enter Marks in Maths:"

read marks_maths

echo "Enter Marks in Physics:"

read marks_physics

# Check if the roll number already exists

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

echo "$rollno:$name:$marks_hindi:$marks_maths:$marks_physics" >> "$log_file"

echo "roll number exists" >> "$log_file"

echo "Roll number exists. Logged in $log_file."

exit 1

fi

# Check if the marks are in the valid range

if [ "$marks_hindi" -lt 1 ] || [ "$marks_hindi" -gt 99 ] || [ "$marks_maths" -lt 1 ] ||
[ "$marks_maths" -gt 99 ] || [ "$marks_physics" -lt 1 ] || [ "$marks_physics" -gt 99 ]; then

echo "$rollno:$name:$marks_hindi:$marks_maths:$marks_physics" >> "$log_file"

echo "marks out of range" >> "$log_file"

echo "Marks out of range. Logged in $log_file."
```

```
exit 1

fi

# Calculate total, percentage, and grade

total=$((marks_hindi + marks_maths + marks_physics))

percentage=$(echo "scale=2; $total / 3" | bc)

grade=$(calculate_grade "$percentage")

# Display the results

echo "Total Marks: $total"

echo "Percentage: $percentage%"

echo "Grade: $grade"

# Append the valid record to the data file

echo
"$rollno:$name:$marks_hindi:$marks_maths:$marks_physics:$total:$percentage:$grad
e" >> "$data_file"
```