PEER LEARNING DOCUMENT

Question - 1

Write a bash script to get the current date, time, username, home directory and current working directory.

Approach -

$(date +"Year: %Y, Month: %m, Day: %d")	Command to fetch the date
$(date +"%T")	Command to fetch the time
$(whoami)	Command to fetch the current working user
$(echo $HOME)	Command to fetch the Home directory
$(pwd)	Command to fetch the current working directory

Code -

```
current_date=$(date +"Year: %Y, Month: %m, Day: %d") # Command to fetch the date
current_time=$(date +"%T") # Command to fetch the time
current_user=$(whoami) # Command to fetch the current working user
home_directory=$(echo $HOME) # Command to fetch the Home directory
current_directory=$(pwd) # Command to fetch the current wokring directory

# Printing the fetched variables
echo "Current date: $current_date"
echo "Current time: $current_time"
echo "Username: $current_user"
echo "Home directory: $home_directory"
echo "Current working directory: $current_directory"

exit 0
```

Sarthak's Code -

#command to get only date
echo current date : $(date +%F)
#command to get only time
echo current time : $(date +%T)
#command to get Username
echo Username : $(whoami)
#command to get Home directory
echo Home directory: ~
#command to get current working directory
echo Current working directory :$(pwd)

Approach - All command's approach is similar but except there in Fetching date there is change in format. $(date +%F)

Purshottam's Code-

```
#!/bin/bash

#Question 1

#Command to get current date and time
echo "Current date and time is:  $(date)"

#Command to get only date
echo "Date is: $(date +%F)"

#Command to get only time
echo "Time is: $(date +%T)"

#Command to get username
echo "Username: $(whoami)"

#Command to get Home Directory
echo "Home Directory: $HOME"

#Command to get users current working directory
echo "Current Working Directory: $(pwd)"
```

Approach - All command's approach is similar but except there in Fetching date there is change in format. $(date).

Question - 2

Write a bash script (name Table.sh) to print the Table of a number by using a while loop. It should support the following requirements.

- **The script should accept the input from the command line.**
- **If you don't input any data, then display an error message to execute the script correctly.**

Approach -

We verify whether any arguments have been passed, and if not, we issue an error and quit the programme with exit status 1. If not, we run a loop to print all the arguments passed, then another while loop on all the arguments, start a counter internally on the second while loop, increment it after each iteration, fetch the result, and repeat this internal loop for each element in the arguments list until the counter value is less than 10, at which point the programme ends.

Code -

```
# If no arguments are passed the raise an error.
if [ $# -eq 0 ]; then
  echo "Error, Please enter a argument to generate table"
  exit 1
fi

echo "No of number passed as arguments - $#"
i=1;
for number in "$@"
do
    echo "Argument - $i: $number";
    i=$((i + 1));
done
echo ""
i=1;
j=$#;
while [ $i -le $j ]
do
        n=$1 # Intiliaing the first argument as n
        c=1 # Counter Variable
        echo "Table of $n:"
        # Using while loop to generate the table
        while [ $c -le 10 ] # while counter is less than 10
        do
          result=$(( $n * $c )) # Calculating the product
          echo "$n x $c = $result" # Printing the product
          c=$(( $c + 1 )) # Incrementing the counter
        done
    shift 1;
        i=$((i+1))
        echo ""
done

```

Sarthak's Code -

echo "Enter the number -"
#Taking user input
read n
Checking through case statement
case $n in
To check if it is a valid number or not
[^0-9])

```
            echo "Please provide a valid input"
  ;;
# Regex to check for number
[0-9]*)
i=1
while [ $i -le 10 ]
do
res=`expr $i \* $n`
echo "$n * $i = $res"
((++i))
done
;;
# condition to check for no input
*)
            echo "Error !Please provide input"
            ;;
esac
```

Approach - Sarthak's Code is a little bit different from mine. Check for whether the input is correct or not .

Purushottam 's code -

```
#!/bin/bash


#Question 2

#Taking user input
echo "Enter the number: "
read n


# Checking through case statement
case $n in
#Regular expression to check if the given input is not a number. If satisfies it results in a invalid input
*[^0-9]*)
     echo "Please input only number!! $n is not a number"
     ;;
#Regex to check for number
[0-9]*)
    i=1

    while [ $i -le 10 ]
```

```
    do
    res=`expr $i \* $n`

    echo "$n * $i = $res"

   (( ++i))

    done
    ;;
#Condition that will handle the situation when there is no input
*)
    echo "Error!! Please Provide Input To Get The Table"
    ;;
esac
```

Approach - Purushottam's Code is a little bit different from mine. Check for whether the input is correct or not.

Question - 3

Write a Function in bash script to check if the number is prime or not? It should support the following requirement. - The script should accept the input from the User.

Approach -

We created a function called is prime to determine whether a number is prime or not. If the number is less than 2, we indicate that it is not a prime, and if not, we run a loop from 2 to number/2+1 to see if any of the above numbers divide the given number. If we find any numbers, we can conclude that the number is not a prime because it has a divisor other than 1 and itself.

We receive user input in the main code, store it in a variable, and then call the function while sending the argument as command line arguments.

Code -

```bash
function is_prime() {
  num=$1 # Intilizing num by the first argument
  # If num is less than 2 it is not prime
  if [ $num -lt 2 ]; then
    echo "$num is not a prime number."
    return
  fi
  # Loop from 2 to num/2 to verify its divisors
  for (( i=2; i<$((num/2+1)); i++ ))
  do
    if [ $(($num%$i)) -eq 0 ]; then # if found it is not prime numer
      echo "$num is not a prime number."
      return
    fi
  done
  # else it is prime
  echo "$num is a prime number."
}
# Reading input from user and storing in num
read -p "Enter a number: " num
# Passing it to function is_prime
is_prime $num

exit 0
```

Sarthak's Code -

```
echo "Enter the number -"
#Taking the input
read n
i=2
#flag , it will change into 1 if n will be not a prime number
flag=0
while test $i -le `expr $n / 2`
do
if test `expr $n % $i` -eq 0
then
flag=1
fi
i=`expr $i + 1`
done
if test $n -eq 1
then
```

```
echo "The number is Not Prime"
elif test $flag -eq 1
then
echo "The number is Not Prime"
else
echo "The number is Prime"
fi
```

Approach - Sarthak's code approach is similar

Purushottam's Code -

```
#!/bin/bash


#Taking user input
echo "Enter a number to check for Prime or not"
read number

#Function to check if a number is prime or not
check_prime(){
  count=0
  num=$1
  for (( i=2 ; i<=$num/2 ;i++ ));
  do
    if [ `expr $num % $i` -eq 0 ]
    then
       count=1
    fi
  done
  if ([ $count -eq 1 ] || [ $num -eq 1 ])
  then
     echo "The given number $num is not a prime number "
  else
     echo "The given number $num is a prime number"
  fi
}

#Checking through case statement if a number is valid or not.If not valid then display the error message otherwise call the check_prime function to display whether number is prime or not.If there is no input,it will display
case $number in
*[^0-9]*)
     echo "Please Enter a Valid Number!! $number is not a number"
```

```
    ;;
[0-9]*)
        echo $(check_prime $number)
        ;;
*)
        echo "Error!! Please Provide Input"
        ;;
esac
```

Approach - Purushottam's code approach is similar

Question - 4

Create a bash script that supports the following requirement.

- **Create a folder 'Assignment'.**
- **Create a file 'File1.txt' inside 'Assignment' Folder.**
- **Copy all the content of Table.sh(2nd script) in 'File1.txt' without using 'cp' and 'mv' command.**
- **Append the text Welcome to Sigmoid' to the 'File1.txt' file.**
- **List all the directories and files present inside Desktop Folder.**

Approach -

We have used the following commands to meet the desired requirements and the function of command is as follows.

Command		Function
mkdir ~/Desktop/Assignment	-	Creating folder using mkdir
touch ~/Desktop/Assignment/File1.txt	-	Creating file using touch
cat ~/Desktop/Table.sh >> ~/Desktop/Assignment/File1.txt	-	Copying data in q2 to file1 using cat
echo "Welcome to Sigmoid" >> ~/Desktop/Assignment/File1.txt	-	Appending given text to file1
ls -la ~/Desktop/	-	Printing files and folders in Desktop

Code

```
# Creating folder using mkdir
mkdir ~/Desktop/Assignment
echo "Created Assignment Folder"
# Creating file using touch
touch ~/Desktop/Assignment/File1.txt
echo "Created File1.txt in Assignment Folder"
# Copying data in q2 to file1 using cat
cat ~/Desktop/Table.sh  >> ~/Desktop/Assignment/File1.txt
echo "Data in q2.sh copied to File1.txt using cat command"
# Appending given text to file1
echo "Welcome to Sigmoid" >> ~/Desktop/Assignment/File1.txt

echo "Folders in the Desktop"
# Printing files and folders in Desktop
ls -la ~/Desktop/

exit 0
```

Sarthak's Code -

mkdir Assignment
touch Assignment/File1.txt
cat Table.sh >> Assignment/File1.txt
echo " Welcome to Sigmoid" >> Assignment/File1.txt
ls -l ~/Desktop

Approach - Sarthak's approach is similar to mine.

Purushottam's Code -

#!/bin/bash

#Question 4

Command for creating the Assignment folder
mkdir Assignment

Command for creating the file "File1.txt" inside Assignment folder
touch Assignment/File1.txt

#Command for copying the content of table.sh in File1.txt
cat Table.sh > Assignment/File1.txt

# Commands to append the text("Welcome to Sigmoid") in File1.txt
str="Welcome to Sigmoid"
echo $str >> Assignment/File1.txt

Listing all the directories and files present inside Desktop Folder and appending it to another text file
ls -al ../ >> DesktopListDirectories.txt
ls -al ~/Desktop >> DesktopListDirectories.txt

echo "Created a file DesktopListDirectories.txt containing all the list and directories present inside Desktop folder"

#Command to open the above created text file containing list of all files and directories present inside Desktop folder
open DesktopListDirectories.txt

Approach - Purushottam's approach is similar to mine.

Question - 5

You have given an array. Using Bash script, print its length, maximum element and minimum element. arr=(2 3 4 1 6 7).

Code -

```
arr=( 2 3 4 1 6 7)
echo "Length of the array - "
echo ${#arr[@]}   # Length of the array
# Method - 1
# Using sort function to find the max and min element.
echo "Max and Min of the array using sort function"
IFS=$'\n'
echo "Maximum in the array - "
echo "${arr[*]}" | sort -nr | head -n1 #Sorting in reverse and fetch the first element
echo "Minimum in the array - "
echo "${arr[*]}" | sort -n | head -n1 # Sorting and fetch the first element

# Method - 2
# Using for loop to find the max and min element.
echo "Max and Min of the array using for loop"
max=${arr[0]} #Assuming first element as max
# Travesing every element in the loop if any element is more than the assumed one change max to it or else continue traversing.
for n in "${arr[@]}" ; do
    if [ $n -ge $max ]; then
                max=$n
        fi
done
echo "Maximum in the array - $max"
min=${arr[0]} #Assuming first element as min
# Travesing every element in the loop if any element is less than the assumed one change min to it or else continue traversing.
for n in "${arr[@]}" ; do
    if [ $n -le $min ]; then
        min=$n
        fi
done
echo "Minimum in the array - $min"
```

Sarthak's Code -

Taking all the element of the array
read -a integers

```
# we assume here that biggest and smallest element of the array is the first value of the array
biggest=${integers[0]}
smallest=${integers[0]}
for i in ${integers[@]}
do
   if [[ $i -gt $biggest ]]  # condition for biggest value
   then
      biggest="$i"
   fi
   if [[ $i -lt $smallest ]]  # condition for smallest value
   then
      smallest="$i"
   fi
done
echo "The length of the array is ${#integers[@]}"
echo "The largest number is $biggest"
echo "The smallest number is $smallest"
```

Approach - Sathrak's Approach his code takes input as an array then it has two variables biggest and smallest and traverses the array appropriately and applies the condition.

Purushottam's Code -

```
#!/bin/bash

#Question 5

#Taking the user input for size of an array
echo "Enter the size of array: "
read size


# Taking User Input From shell for an array
#Assigning an empty array
arr=()
for(( i=0; i<size ;i++ ));
do
        echo "Enter $(($i+1)) element"
        read n
        arr[$i]=$n
done

echo "Total Number of elements: ${#arr[@]}"
```

```
echo "The array elements are: "
echo ${arr[@]}

# Function to find maximum and minimum element in a given array
max_min_ele(){
        max=${arr[0]}
        min=${arr[0]}

        for ele in "${arr[@]}";
        do
          if [ $ele -gt $max ]
          then
            max=$ele
          fi
          if [ $ele -lt $min ]
          then
                  min=$ele
          fi
        done
        echo "Maximum element in an array: $max"
        echo "Minimum element in an array: $min"
}

#Calling the above function max_min_ele
max_min_ele
```

Approach - Purushottam's Approach his code takes input as an array then it has two variables biggest and smallest and traverses the array appropriately and applies the condition.