PEER LEARNING DOCUMENT
(COMMAND LINE)

My peers :
1) **Pururshottam Dubey**
2) **Atul Singh**

My solution explanation:

Question 1:
Write a bash script to get the current date, time, username, home directory and current working directory.

```
echo Answer 1-
#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)
```

Explanation:
In the above solution echo command is used which is used to display whatever is written after that command . The keyword date is used to get the date , whoami is used to fetch the user name, the tilde (~) is used to get the home directory and working directory using pwd.
The dollar sign $ is used to signify that whatever written after that is treated as a command not the input to echo command.

Purushottam and Atul had a similar approach for this question.

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.

#answer 2:
My solution explanation:

```
        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
```

Explanation:
This is a shell script that prompts the user to enter a number and then uses a case statement to check the input. If the input is not a valid number, it displays an error message. If the input is a valid number, it uses a loop to print the multiplication table of that number up to 10.

Here's a step-by-step breakdown of what the script does:

- The first line displays the message "Enter the number -" to prompt the user to enter a number.
- The second line reads the user's input and stores it in the variable n.

- The case statement begins, checking the value of n.
- The first case checks if n contains any characters other than digits using a regular expression *[^0-9]*. If it does, it displays the message "Please provide a valid input" and exits the script.
- The second case checks if n contains only digits using the regular expression [0-9]*. If it does, it sets the variable i to 1 and enters a loop that will run 10 times.
- After the loop completes, the script exits the case statement.
- If n did not match any of the previous cases, the script displays the message "Error! Please provide input" and exits.

Overall, this script allows the user to enter a number and prints its multiplication table up to 10, or displays an error message if the input is invalid.

Purushottam and Atul had a similar approach for this question. They have just used different types of loops and different variables.

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

#answer 3
My solution explanation:

```
        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
```

Explanation:

This is a Bash script that takes an input number from the user and checks if it is a prime number or not. Here is how the script works:

- The script prompts the user to enter a number.
- The script initializes a counter variable i to 2 and a flag variable flag to 0.
- The script enters a while loop that continues until i is less than or equal to half of the input number.
- Inside the while loop, the script checks if the input number is divisible by i (i.e., if the input number modulo i equals 0). If it is, the script sets the flag variable to 1 to indicate that the input number is not prime.
- The script increments the counter variable i by 1 and continues the loop.
- After the loop has completed, the script checks if the input number is equal to 1. If it is, the script prints "The number is Not Prime" because 1 is not a prime number.
- If the input number is not equal to 1, the script checks the value of the flag variable. If it is equal to 1, the script prints "The number is Not Prime" because the input number is not prime. If the flag variable is equal to 0, the script prints "The number is Prime" because the input number is prime.

Note that this script assumes that the input number is a positive integer. If the user enters a non-integer or a negative number, the script may not behave as expected.

Purushottam and Atul had a similar approach for this question. They have just used different types of loops and different variables.

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.

#answer 4:
My solution explanation:

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

Explanation:

- mkdir Assignment: creates a new directory named "Assignment".
- touch Assignment/File1.txt: creates a new file named "File1.txt" inside the "Assignment" directory.
- cat Table.sh >> Assignment/File1.txt: appends the contents of a file named "Table.sh" to the end of the "File1.txt" file.
- echo " Welcome to Sigmoid" >> Assignment/File1.txt: appends the text "Welcome to Sigmoid" (with two leading spaces) to the end of the "File1.txt" file.
- ls -l ~/Desktop: lists the contents of the Desktop directory in long format, showing detailed information about each file and directory. The tilde character (~) represents the current user's home directory.

Purushottam and Atul had a similar approach for this question. Since the commands are straight forward so we all had the same answer for this question.

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).

#answer 5:
My solution explanation:

# 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"
```

Explanation:

This is a bash script that reads input integers as an array and finds the largest and smallest element in the array. The code initializes the variables "biggest" and "smallest" to the first element of the array and then iterates through the array using a for loop.
During each iteration, the code compares the current element with the current value of "biggest" and "smallest" using if statements. If the current element is greater than the current value of "biggest", the code updates the value of "biggest" to the current element.
After the loop, the code prints the length of the array, as well as the values of "biggest" and "smallest".

Purushottam and Atul had a similar approach for this question.