Assignment 2

Part A

What will the following commands do?

- **echo "Hello, World!"**

The echo command in the shell is used to display a line of text or a variable's value to the terminal.

```
cdac@Farah:~$ echo "Enter Name"
Enter Name
```

```
cdac@Farah:~$ name="Alina"
echo "Hello, $name!"
Hello, Alina!
cdac@Farah:~$ name=Alina
echo "Hello", $name!
Hello, Alina!
cdac@Farah:~$
```

- **name="Productive"**

name is a variable and Productive is the assigned value to it.

- **touch file.txt**

Used to create file

```
cdac@Farah:~$ touch file.txt
cdac@Farah:~$ ls
LinuxAssignment  Problem2  file.txt
```

- **ls -a**

Display all the files and directories of that particular directory including hidden files also that start with dot(.)

```
cdac@Farah:~$ ls -a
.   .bash_history   .bashrc   .local       .profile                  LinuxAssignment  file.txt
..  .bash_logout    .cache    .motd_shown   .sudo_as_admin_successful  Problem2
```

- **rm file.txt**

Removes the particular file.

```
cdac@Farah:~$ touch file.txt
cdac@Farah:~$ ls
LinuxAssignment  Problem2  file.txt
```

```
cdac@Farah:~$ rm file.txt
cdac@Farah:~$ ls
LinuxAssignment  Problem2
```

- **cp file1.txt file2.txt**

Creating 2 files named file1 and file2 respectively.

```
cdac@Farah:~$ touch file1.txt
cdac@Farah:~$ touch file2.txt
```

Edited file1 and displaying the content.

```
cdac@Farah:~$ nano file1.txt
cdac@Farah:~$ cat file1.txt
Hello!
```

Copying file1 content to existing file2.

```
cdac@Farah:~$ cp file1.txt file2.txt
cdac@Farah:~$ cat file2.txt
Hello!
```

File3 doesnot exist so cp command is creating file3 and then copying the content.

```
cdac@Farah:~$ cp file1.txt file3.txt
cdac@Farah:~$ cat file3.txt
Hello!
```

- **mv file.txt /path/to/directory/**

Moving the file from current location to specified directory.

```
cdac@Farah:~$ ls
LinuxAssignment  Problem2  file.txt  file1.txt  file2.txt  file3.txt
cdac@Farah:~$ mv file.txt /home/cdac/LinuxAssignment/
cdac@Farah:~$ ls
LinuxAssignment  Problem2  file1.txt  file2.txt  file3.txt
cdac@Farah:~$ cd LinuxAssignment
cdac@Farah:~/LinuxAssignment$ ls
Docs  file.txt  file1.txt
```

- **chmod 755 script.sh**

```
cdac@Farah:~/LinuxAssignment$ touch script.sh
cdac@Farah:~/LinuxAssignment$ ls -l
total 12
drwxr-xr-x 2 cdac cdac 4096 Aug 30 14:02 Docs
---------- 1 cdac cdac    6 Aug 30 15:38 file.txt
-rw-r--r-- 1 cdac cdac   53 Aug 30 14:28 file1.txt
-rw-r--r-- 1 cdac cdac    0 Aug 30 18:25 script.sh
cdac@Farah:~/LinuxAssignment$ chmod 755 script.sh
cdac@Farah:~/LinuxAssignment$ ls -l
total 12
drwxr-xr-x 2 cdac cdac 4096 Aug 30 14:02 Docs
---------- 1 cdac cdac    6 Aug 30 15:38 file.txt
-rw-r--r-- 1 cdac cdac   53 Aug 30 14:28 file1.txt
-rwxr-xr-x 1 cdac cdac    0 Aug 30 18:25 script.sh
```

- **grep "pattern" file.txt**

```
cdac@Farah:~/LinuxAssignment$ cat file1.txt
Hello!
Ni Hao!
Hola!
Salaam!
Namaste!
Bonjour!
Ciao!
cdac@Farah:~/LinuxAssignment$ grep "Hola!" file1.txt
Hola!
```

- **kill PID**

Used to terminate the process.

- **mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt**

- **ls -l | grep ".txt"**

Filter out the .txt file

```
cdac@Farah:~/LinuxAssignment$ ls -l | grep ".txt"
---x------ 1 cdac cdac    6 Aug 30 15:38 file.txt
-rwxr--r-- 1 cdac cdac   53 Aug 30 14:28 file1.txt
```

- **cat file.txt file1.txt | sort | uniq**

```
cdac@Farah:~/LinuxAssignment$ cat file1.txt
Hello!
Ni Hao!
Hola!
Salaam!
Namaste!
Bonjour!
Ciao!
cdac@Farah:~/LinuxAssignment$ nano file.txt
cdac@Farah:~/LinuxAssignment$ cat file.txt
Hello!
Salaam!
Hola!
Bye!
Bonjour!
Ciao!

cdac@Farah:~/LinuxAssignment$ cat file.txt file1.txt | sort | uniq

Bonjour!
Bye!
Ciao!
Hello!
Hola!
Namaste!
Ni Hao!
Salaam!
```

- **ls -l | grep "^d"**

```
cdac@Farah:~/LinuxAssignment$ ls -l | grep "^d"
drwxr-xr-x 2 cdac cdac 4096 Aug 30 14:02 Docs
```

- **grep -r "pattern" /path/to/directory/**

```
cdac@Farah:~/LinuxAssignment$ grep -r "Hello" /home/cdac/LinuxAssignment/
/home/cdac/LinuxAssignment/file1.txt:Hello!
/home/cdac/LinuxAssignment/file.txt:Hello!
/home/cdac/LinuxAssignment/Docs/file2.txt:Hello!
```

- **cat file1.txt file2.txt | sort | uniq -d**

-d : Displays only duplicate lines.

```
cdac@Farah:~/LinuxAssignment$ cat file.txt
Hello!
Salaam!
Hola!
Bye!
Bonjour!
Ciao!

cdac@Farah:~/LinuxAssignment$ cat file1.txt
Hello!
Ni Hao!
Hola!
Salaam!
Namaste!
Bonjour!
Ciao!
cdac@Farah:~/LinuxAssignment$ cat file.txt file1.txt | sort | uniq -d
Bonjour!
Ciao!
Hello!
Hola!
Salaam!
```

- **chmod 644 file.txt**

```
cdac@Farah:~/LinuxAssignment$ chmod 644 script.sh
cdac@Farah:~/LinuxAssignment$ ls -l
total 12
drwxr-xr-x 2 cdac cdac 4096 Aug 30 14:02 Docs
---------- 1 cdac cdac    6 Aug 30 15:38 file.txt
-rw-r--r-- 1 cdac cdac   53 Aug 30 14:28 file1.txt
-rw-r--r-- 1 cdac cdac    0 Aug 30 18:25 script.sh
```

- **cp -r source_directory destination_directory**

cp -r file.txt file1.txt - Copies directories and their contents recursively.

```
cdac@Farah:~/LinuxAssignment$ cat file.txt
Hello!
Salaam!
Hola!
Bye!
Bonjour!
Ciao!
```

```
cdac@Farah:~/LinuxAssignment$ cat file1.txt
Hello!
Salaam!
Hola!
Bye!
Bonjour!
Ciao!
```

- **find /path/to/search -name "*.txt"**
- **chmod u+x file.txt**

```
cdac@Farah:~/LinuxAssignment$ chmod u+x file.txt
cdac@Farah:~/LinuxAssignment$ ls -l
total 12
drwxr-xr-x 2 cdac cdac 4096 Aug 30 14:02 Docs
---x------ 1 cdac cdac    6 Aug 30 15:38 file.txt
-rw-r--r-- 1 cdac cdac   53 Aug 30 14:28 file1.txt
-rw-rw-rw- 1 cdac cdac    0 Aug 30 18:25 script.sh
cdac@Farah:~/LinuxAssignment$ chmod u+x file1.txt
cdac@Farah:~/LinuxAssignment$ ls -l
total 12
drwxr-xr-x 2 cdac cdac 4096 Aug 30 14:02 Docs
---x------ 1 cdac cdac    6 Aug 30 15:38 file.txt
-rwxr--r-- 1 cdac cdac   53 Aug 30 14:28 file1.txt
-rw-rw-rw- 1 cdac cdac    0 Aug 30 18:25 script.sh
```

- **echo $PATH**

Part B

Identify True or False:

1. ls is used to list files and directories in a directory. **- True.**

2. mv is used to move files and directories. **- True.**

3. cd is used to copy files and directories. - **False.**

4. pwd stands for "print working directory" and displays the current directory. **- True.**

5. grep is used to search for patterns in files. **- True.**

6. chmod 755 file.txt gives read, write, and execute permissions to the owner, and read and execute permissions to group and others. **- True.**

7. mkdir -p directory1/directory2 creates nested directories, creating directory2 inside directory1 if directory1 does not exist. **- True.**

8. rm -rf file.txt deletes a file forcefully without confirmation. **- True.**

Identify the Incorrect Commands:

1. **chmodx** is used to change file permissions. **– Incorrect**
    - **chmod** is the correct command.

2. **cpy** is used to copy files and directories. **– Incorrect**
    - **cp** is the correct command to copy.

3. **mkfile** is used to create a new file. **– Incorrect**
    - **touch** command is used to create a new file or nano command is used to create a new file and add content to it and edit later also.

4. **catx** is used to concatenate files. **– Incorrect**
    - **cat** is the correct command. It is used to concatenate files.

5. **rn** is used to rename files. **– Incorrect**
    - **mv** command is used to rename files.

Part C

Question 1: Write a shell script that prints "Hello, World!" to the terminal.

Created a new shell script file.

```
cdac@Farah:~$ nano shellFile.sh
```

#!/bin/bash - to specify the script interpreter. Used echo to display.

```
  GNU nano 6.2
#!/bin/bash
echo Hello World
```

Getting the output.

```
cdac@Farah:~$ bash shellFile.sh
Hello World
```

Question 2: Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable.

```
  GNU nano 6.2
#!/bin/bash
name="CDAC Mumbai"
echo $name
```

```
cdac@Farah:~$ nano shellScript.sh
cdac@Farah:~$ ls -l
total 32
drwxr-xr-x 3 cdac cdac 4096 Aug 30 18:25 LinuxAssignment
drwxr-xr-x 2 cdac cdac 4096 Aug 29 14:01 Problem2
-rwxr-xr-x 1 cdac cdac    7 Aug 30 15:39 file1.txt
-rw-r--r-- 1 cdac cdac    7 Aug 30 15:39 file2.txt
-rw-r--r-- 1 cdac cdac    7 Aug 30 15:40 file3.txt
-rw-r--r-- 1 cdac cdac   58 Aug 30 20:06 shellFile.s
-rwxr-xr-x 1 cdac cdac   44 Aug 30 20:10 shellFile.sh
-rw-r--r-- 1 cdac cdac   42 Aug 30 20:13 shellScript.sh
cdac@Farah:~$ chmod +x shellScript.sh
cdac@Farah:~$ ls -l
total 32
drwxr-xr-x 3 cdac cdac 4096 Aug 30 18:25 LinuxAssignment
drwxr-xr-x 2 cdac cdac 4096 Aug 29 14:01 Problem2
-rwxr-xr-x 1 cdac cdac    7 Aug 30 15:39 file1.txt
-rw-r--r-- 1 cdac cdac    7 Aug 30 15:39 file2.txt
-rw-r--r-- 1 cdac cdac    7 Aug 30 15:40 file3.txt
-rw-r--r-- 1 cdac cdac   58 Aug 30 20:06 shellFile.s
-rwxr-xr-x 1 cdac cdac   44 Aug 30 20:10 shellFile.sh
-rwxr-xr-x 1 cdac cdac   42 Aug 30 20:13 shellScript.sh
cdac@Farah:~$ bash shellScript.sh
CDAC Mumbai
```

Question 3: Write a shell script that takes a number as input from the user and prints it.

```
#!/bin/bash
echo enter a number
read num
echo "Number is" $num
```

```
cdac@Farah:~$ bash shellScript.sh
enter a number
24
Number is 24
```

Question 4: Write a shell script that performs addition of two numbers (e.g., 5 and 3) and prints the result.

Two ways to print sum of numbers:-

```
#!/bin/bash
echo Enter Num
read Num1
echo "Enter Num2";
read Num2
echo "Sum is" $((Num1+Num2))
echo "Sum is" $(expr $Num1 + $Num2)
```

```
cdac@Farah:~$ bash shellScript.sh
Enter Num
10
Enter Num2
24
Sum is 34
Sum is 34
```

Question 5: Write a shell script that takes a number as input and prints "Even" if it is even, otherwise prints "Odd".

```
#!/bin/bash
echo Enter Num
read Num
if [ $((Num % 2)) -eq 0 ];
then
        echo "Even"
else
        echo "Odd"
fi
```

```
cdac@Farah:~$ bash shellScript.sh
Enter Num
13
Odd
```

Question 6: Write a shell script that uses a for loop to print numbers from 1 to 5.

```
#!/bin/bash
for (( i=1; i<=5; i++ ))
do
        echo Number $i
done
```

```
cdac@Farah:~$ bash shellScript.sh
Number 1
Number 2
Number 3
Number 4
Number 5
```

Question 7: Write a shell script that uses a while loop to print numbers from 1 to 5.

```
#!/bin/bash
i=1
while (( i<=5 ))
do
        echo Number $i
        (( i++ ))
done
```

```
cdac@Farah:~$ bash shellScript.sh
Number 1
Number 2
Number 3
Number 4
Number 5
```

Question 8: Write a shell script that checks if a file named "file.txt" exists in the current directory. If it does, print "File exists", otherwise, print "File does not exist".

```
#!/bin/bash

filename="/home/cdac/file1.txt"

if [ -f "$filename" ];
then
        echo Regular file exists.
else
        echo file does not exists.
fi
```

```
cdac@Farah:~$ bash shellScript.sh
Regular file exists.
```

[-f "$filename"]: Checks if the file is a regular file.

[-e "$filename"]: Checks if the file exists (any type of file).

[-x "$file"]: Checks if the file is executable.

[-d "$directory"]: Checks if the path is a directory.

```
#!/bin/bash

filename="/home/cdac/Problem2"

if [ -d "$filename" ];
then
        echo Directory exists.
else
        echo Directory does not exists.
fi
```

Question 9: Write a shell script that uses the if statement to check if a number is greater than 10 and prints a message accordingly.

```
#!/bin/bash

echo Enter a Number
read Num1

Num2=10

if [ $Num1 -gt $Num2 ]

then
        echo Number is greater than 10.
else
        echo Number is not greater than 10.
fi
```

```
cdac@Farah:~$ bash shellScript.sh
Enter a Number
11
Number is greater than 10.
```

Don't give space Num2 = 0 like this. It will give error.

```
Num2 = 10
```

Question 10: Write a shell script that uses nested for loops to print a multiplication table for numbers from 1 to 5. The output should be formatted nicely, with each row representing a number and each column representing the multiplication result for that number.

Question 11: Write a shell script that uses a while loop to read numbers from the user until the user enters a negative number. For each positive number entered, print its square. Use the break statement to exit the loop when a negative number is entered.

```
#!/bin/bash

echo Enter a Number
read Num
while((Num>=0))
do
        echo $((Num*Num))
break
done
```

```
cdac@Farah:~$ bash positive.txt
Enter a Number
4
16
```

Part E

1. Consider the following processes with arrival times and burst times:

Process	Arrival Time	Burst Time
P1	0	5
P2	1	3
P3	2	6

Calculate the average waiting time using First-Come, First-Served (FCFS) scheduling.

Solution:-

First-Served (FCFS) Scheduling -

Process	Arriving Time	Burst Time	Waiting Time	
P1	0	5	0	<--- P1 = 0
P2	1	3	4	<--- P2 = 5-1=4
P3	2	6	6	<--- P3 = 8-2 =6
	P1	P2	P3	
0	5	8	14	<--- CPU Alloacted Time
	Waiting Time =	CPU Alloacted Time - Arrival Time		
	Average Waiting Time = (0 + 4 + 6)/3			
	AWT = 10/3			
	AWT = 3.33			

2. Consider the following processes with arrival times and burst times:

Process	Arrival Time	Burst Time
P1	0	3
P2	1	5
P3	2	1
P4	3	4

Calculate the average turnaround time using Shortest Job First (SJF) scheduling.

Solution:-

Shortest Job First (SJF) Scheduling –

Process	Arriving Time	Burst Time	Waiting Time	TurnAround Time (TAT)
P1	0	3	0	3
P2	1	5	7	12
P3	2	1	1	2
P4	3	4	1	5
				TAT = 5.5

	P1	P4	P2	P3
0	4	6	12	20

Waiting Time = CPU Allocation Time - Arrival Time

TAT = Completion Time - Arrival Time

Here Completion Time for every Process is -
P1 = 4, P2 = 12,

Average TAT Time = (3 + 12 + 2 + 5)/4
TAT = 22/4
TAT = 5.5

3. Consider the following processes with arrival times, burst times, and priorities (lower number indicates higher priority):

Process	Arrival Time	Burst Time	Priority
P1	0	6	3
P2	1	4	1
P3	2	7	4
P4	3	2	2

Calculate the average waiting time using Priority Scheduling.

Solution:-

Priority Scheduling -

Process	Arrival Time	Burst Time	Priority	TurnAround Time (TAT)	CT	Waiting Time
P1	0	6	3	6	6	0
P2	1	4	1	9	10	5
P3	2	7	4	14	16	7
P4	3	2	2	9	12	7
						AVG WT = 4.75

	P1	P2	P4	P3
0	6	10	12	16

Waiting Time = CPU Allocation Time - Arrival Time

TAT = Completion Time - Arrival Time

Avg WT = (0 + 5 + 5 + 7)/4

Avg WT = 17/4

Avg WT =4.5

4. Consider the following processes with arrival times and burst times, and the time quantum for Round Robin scheduling is 2 units:

Process	Arrival Time	Burst Time
P1	0	4
P2	1	5
P3	2	2
P4	3	3

Calculate the average turnaround time using Round Robin scheduling.

Solution:-

Round Robin Scheduling –

Process	Arrival Time	Burst Time	CT	TAT	Waiting Time			
P1	0	4	10	10	6			
P2	1	5	14	13	8			
P3	2	0	6	4	2			
P4	3	3	13	10	7			

	P1	P2	P3	P4	P1	P2	P3	P4
0	2	4	6	8	10	12	14	16

Waiting Time = CPU Allocation Time - Arrival Time

TAT = Completion Time - Arrival Time

TAT= (10 + 13 + 4 + 10)/4

TAT= 37/4

TAT = 9.25