Part A

1. **echo “Hello World!”** will print **Hello World!**

2. **name=”Productive”** will assign the variable **name** the string **Productive**.

3. **touch file.txt** will create a file name **file.txt**.

4. **ls –a** will list all files and directories, including hidden ones (those starting with “**.**”).

5. **rm file.txt** will remove the file **file.txt**.

6. **cp file1.txt file2.txt** will copy and paste the contents of **file1.txt** to **file2.txt**.

7. **mv file.txt /path/to/directory** will move the **file.txt** to the specified path.

8. **chmod 755 script.sh** will change the permission of **script.sh** such that the owner can read, write and execute, the group and other users can read and execute the file.

9. **grep “pattern” file.txt** will search the file **file.txt** for the word **pattern**.

10. **kill PID** will terminate the process with the specified process ID.

11. **mkdir mydir && cd mydir && touch file.txt && echo “Hello, World!” > file.txt && cat file.txt** will create a directory named **mydir**, enter it, create **file.txt**, write **"Hello, World!"** into it, and then display its content.

12. **ls –l | grep “.txt”** will list the file long format and filter the files with **.txt** in its name.

13. **cat file1.txt file2.txt | sort | uniq** will concatenate the files **file1.txt** and **file2.txt** and sort the combine content and remove duplicates.

14. **grep –r “pattern” path/to/directory** will search recursively the word **pattern** inside all the files along given path of the directory.

15. **cat file1.txt file2.txt | sort | uniq –d** will concatenate the files **file1.txt** and **file2.txt** and sort the combined content and keep duplicate lines one of each.

16. **ls –l | grep “^d”** will list only directories in the current directory.

17. **chmod 644 file.txt** will change the permission of the file **file.txt** such that the owner can read and write, and the group and others can read only.

18. **cp –r source_directory destination_directory** will recursively copy **source_directory** to **destination_directory** including all files and subdirectories.

19. **find /path/to/search –name "*.txt"** will search .**txt** files within the given path and its subdirectories.

20. **chmod u+x file.txt** will change the permission of owner to allow the owner the right to execute the file **file.txt**.

21. **echo $PATH** will display e system's PATH environment variable, which lists directories where executable files are searched for.

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. **cd** is used for moving through directories and **cp** is used to copy file.
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. Correct command is **chmod**
2. **cpy** is used to copy files and directories.
   Incorrect. Correct command is **cp**.
3. **mkfile** is used to create a new file.
   Incorrect. Correct command is **touch**.
4. **catx** is used to concatenate files.
   Incorrect. Correct command is **cat**.
5. **rn** is used to rename files.
   Incorrect. Correct command is **mv**.

Part C

1. **echo ”Hello, World!”**

```
cdac@FLASH: ~
cdac@FLASH:~$ echo "Hello, World!"
Hello, World!
cdac@FLASH:~$
```

2. **name = “CDAC Mumbai”**
   echo $name

```
cdac@FLASH: ~/LinuxAssignm
cdac@FLASH:~/LinuxAssignment_2$ name="CDAC Mumbai"
cdac@FLASH:~/LinuxAssignment_2$ echo $name
CDAC Mumbai
cdac@FLASH:~/LinuxAssignment_2$
```

3. **nano numprint**

   <u>Inside Editor</u>:
   echo “Enter a number”
   read num
   echo “Number is $num”

```
cdac@FLASH: ~/LinuxAssignm
cdac@FLASH:~/LinuxAssignment_2$ nano numprint
cdac@FLASH:~/LinuxAssignment_2$ bash numprint
Enter a number
45
Number is 45
cdac@FLASH:~/LinuxAssignment_2$
```

4. **num1=5**
   num2=3
   sum=$((num1 + num2))
   echo $sum

```
cdac@FLASH: ~/LinuxAssignm
cdac@FLASH:~/LinuxAssignment_2$ num1=5
cdac@FLASH:~/LinuxAssignment_2$ num2=3
cdac@FLASH:~/LinuxAssignment_2$ sum=$((num1+num2))
cdac@FLASH:~/LinuxAssignment_2$ echo $sum
8
cdac@FLASH:~/LinuxAssignment_2$
```

5. **nano EvenOdd**

   <u>Inside Editor</u>:

```
num=0
echo "Enter a number"
read num
if (($num % 2 == 0));
then
    echo "Even"
else
    echo "Odd"
fi
```

   bash EvenOdd

```
cdac@FLASH: ~/LinuxAssignm
cdac@FLASH:~/LinuxAssignment_2$ nano EvenOdd
cdac@FLASH:~/LinuxAssignment_2$ bash EvenOdd
Enter a number
4
Even
cdac@FLASH:~/LinuxAssignment_2$
```

6. **nano forloop**

<u>Inside Editor</u>:

```
a=0
for a in 1 2 3 4 5
do
    echo $a
done
```

bash forloop

```
cdac@FLASH: ~/LinuxAssignm
cdac@FLASH:~/LinuxAssignment_2$ nano forloop
cdac@FLASH:~/LinuxAssignment_2$ bash forloop
1
2
3
4
5
cdac@FLASH:~/LinuxAssignment_2$
```

7. **nano whileloop**

<u>Inside Editor</u>:

```
a=1
while [ $a –lt 6 ]
do
    echo $a
    ((a++))
done
```

bash whileloop

```
cdac@FLASH: ~/LinuxAssignm
cdac@FLASH:~/LinuxAssignment_2$ nano whileloop
cdac@FLASH:~/LinuxAssignment_2$ bash whileloop
1
2
3
4
5
cdac@FLASH:~/LinuxAssignment_2$
```

8. **nano file**

<u>Inside Editor</u>:

```
if [ -f “file.txt” ];
then
    echo ”File Exist”
else
   echo “File does not exist”
fi
```

bash file

```
cdac@FLASH: ~/LinuxAssignm
cdac@FLASH:~/LinuxAssignment_2$ nano file
cdac@FLASH:~/LinuxAssignment_2$ bash file
File does not exixt
cdac@FLASH:~/LinuxAssignment_2$
```

9. **nano greaterthan**

<u>Inside Editor</u>:

```
echo “Enter a number”
read num
if [ $num –gt 10 ];
then
    echo “$num is greater than 10”
else
    echo “$num is not greater than 10”
fi
```

bash greaterthan

```
cdac@FLASH: ~/LinuxAssignm
cdac@FLASH:~/LinuxAssignment_2$ nano greaterthan
cdac@FLASH:~/LinuxAssignment_2$ bash greaterthan
Enter a number
12
12 is greater than 10
cdac@FLASH:~/LinuxAssignment_2$
```

10. **nano nestedforloop**

<u>Inside Editor</u>:

```
product=0
for i in 1 2 3 4 5
do
    echo "Table of $i"
    for j in 1 2 3 4 5 6 7 8 9 10
    do
        product=`expr $i \* $j `
        echo "$i * $j" = $product
    done
done
```

bash nesterforloop

```
cdac@FLASH: ~/LinuxAssignm
cdac@FLASH:~/LinuxAssignment_2$ nano nestedforloop
cdac@FLASH:~/LinuxAssignment_2$ bash nestedforloop
Table of 1
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
1 * 5 = 5
1 * 6 = 6
1 * 7 = 7
1 * 8 = 8
1 * 9 = 9
1 * 10 = 10
Table of 2
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20
```

cdac@FLASH: ~/LinuxAssignm

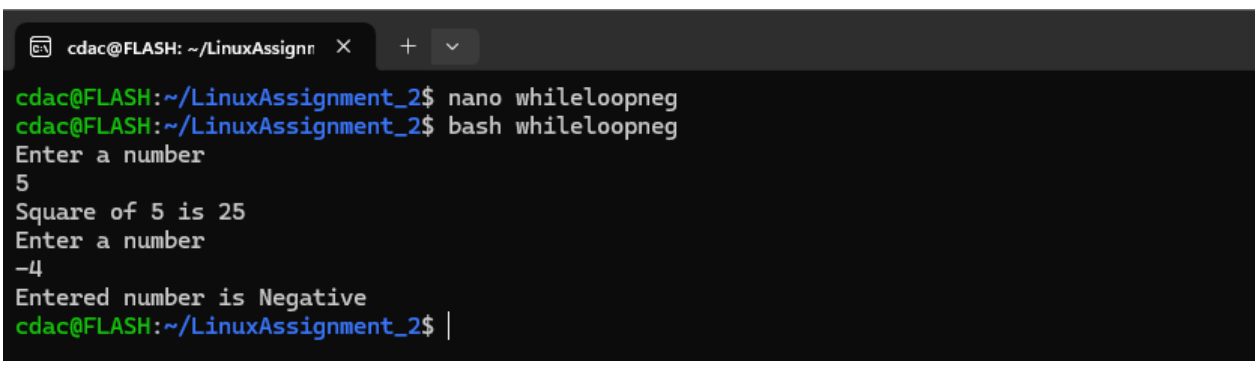
```
Table of 3
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30
Table of 4
4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
4 * 10 = 40
Table of 5
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
cdac@FLASH:~/LinuxAssignment_2$
```

11. **nano whileloopneg**

<u>Inside Editor</u>:

```
while true
do
    echo "Enter a number"
    read num
    if [ $num –lt 0 ];
    then
        echo "Entered number is Negative"
        break
    fi
    echo "Square of $num is $((num * num))"
done
```

bash whileloopneg



Part E

1. **First-Come, First-Served scheduling**

Process ID	Arrival Time	Burst Time	Response Time	Waiting Time
P1	0	5	0	0
P2	1	3	5	4
P3	2	6	14	12

Gantt chart

P1	P2	P3	
0	5	8	14

Average Waiting Time = **5.33 ms**

2. **Shortest Job First Scheduling**

Process ID	Arrival Time	Burst Time	Response Time	Waiting Time	Turnaround Time
P1	0	3	0	0	3
P2	1	5	8	7	12
P3	2	1	2	1	2
P4	3	4	4	1	5

Gantt chart

P1	P1	P3	P1	P4	P2	
0	1	2	3	4	8	13

Average Turnaround Time = **5.5 ms**

3. **Priority Scheduling (Lowest number high priority)**

Process ID	Arrival Time	Burst Time	Priority	Response Time	Waiting Time
P1	0	6	3	0	6
P2	1	4	1	1	1
P3	2	7	4	13	13
P4	3	2	2	5	5

Gantt chart

P1	P2	P2	P2	P4	P1	P3	
0	1	2	3	5	7	13	20

Average Waiting Time = **6.25 ms**

4. **Round Robin with time quantum = 2 units**

Process ID	Arrival Time	Burst Time	Response Time	Waiting Time	Turnaround Time
P1	0	4	0	6	10
P2	1	5	2	10	15
P3	2	2	4	4	6
P4	3	3	6	12	15

Gantt chart: CPU kept idle

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

Average Turnaround Time = **11.5 units**

5. a. Initial the parent program had a variable **x = 5**.
   b. After **fork()** is ran both the parent and child have the variable **x** with value **5**, as a separate copy is created for the child for the same variable.
   c. Now in both cases the value of the variable **x** is incremented by **1**.
   d. Hence we can conclude that the final of **x** in parent as well as child processes after the **fork()** is **6**.