Name : Aditya Anand Bhandare

Part A

1) echo "Hello, World!"
   Ans: echo is a built-in tool that displays text or strings on the terminal
   Output Hello, World!
2) name="Productive"
   Ans: It assigns the value "Productive" to the variable name. This variable can then be used in scripts or commands.
3) touch file.txt
   Ans: Touch used to create empty files and change the access and modification times of existing files .Here empty files will create having name as file.txt
4) ls -a
   Ans: ls - list directory contents,-a do not ignore entries starting with.
   It will gives - list directory contents entries starting with a
5) rm file.txt
   Ans: rm : remove files or directories.It will remove file.txt
6) cp file1.txt file2.txt
   Ans: cp - copy files and directories. This copies file1.txt to file2.txt
7) mv file.txt /path/to/directory/
   Ans: mv move (rename) files
   file.txt is the source file
   /path/to/directory/ is the destination directory
   It move file.txt to the specified directory
8) chmod 755 script.sh
   Ans: is used to change the permissions of script.sh.
   Read, Write, Execute (rwx) 7 Read, Execute (r-x) 5
9) grep "pattern" file.txt

   Ans: grep searches for PATTERNS in each FILE. PATTERNS is one or more patterns separated by newline characters, and grep prints each line that matches a pattern. Typically PATTERNS should be quoted when grep is used in a shell command.

10) kill PID

    Ans : This sends the SIGTERM (signal 15) to allow the process to exit cleanly.

11) mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt
    Ans; mkdir - make directories of name mydir . cd mydir - Changes into the mydir directory
    touch file.txt → Creates an empty file named file.txt.
    echo "Hello, World!" > file.txt → Writes "Hello, World!" into file.txt
    cat file.txt → Displays the contents of file.txt which is Hello, World!
    Output : Hello, World!
12) ls -l | grep ".txt"
    Ans: Lists files in the current directory in long format (showing permissions, owner, size)

| pipe – passes the output
grep ".txt" - Filters and displays only lines that contain .txt, meaning it will show details of files with the .txt extension.

13) cat file1.txt file2.txt | sort | uniq
Ans: cat file1.txt file2.txt- Reads and displays the contents of file1.txt and file2.txt sequentially
Sort- Sorts all lines alphabetically.
uniq - Removes duplicate lines that are now adjacent (after sorting).
This will do sorts, and removes duplicate lines from file1.txt and file2.txt

14) ls -l | grep "^d"
Ans: This command lists only directories in the current directory.

15) grep -r "pattern" /path/to/directory/
Ans: grep - Searches for text in files.
-r (recursive)- Searches inside all subdirectories and files.
"pattern"- The text or regular expression you want to find.
/path/to/directory/ - The directory where the search starts.
This command searches recursively for "pattern" inside all files under /path/to/directory/.

16) cat file1.txt file2.txt | sort | uniq –d
Ans: This command finds duplicate lines that appear in both file1.txt and file2.txt

17) chmod 644 file.txt
Ans: modifies file permissions for file.txt using the chmod.
6 (Owner) -Read & Write (rw-) 4 (Group) - Read-only (r--) 4 (Others) -Read-only (r--)

18) cp -r source_directory destination_directory
Ans : cp -Copy command.
-r (recursive) -Copies directories, subdirectories, and files.
source_directory -The directory you want to copy.
destination_directory - The target location where the directory will be copied.

19) find /path/to/search -name "*.txt"
Ans: This command searches for all .txt files inside /path/to/search (including subdirectories

20) chmod u+x file.txt
Ans

chmod -Change file permissions.
u - User (owner of the file).
+x - Adds execute permission.
This adds execute (x) permission to the user (owner) of file.txt.

21) echo $PATH
Ans: The echo $PATH command displays the system's PATH environment variable, which defines directories where the system looks for executable files. $PATH is a environment variable that is file location-related

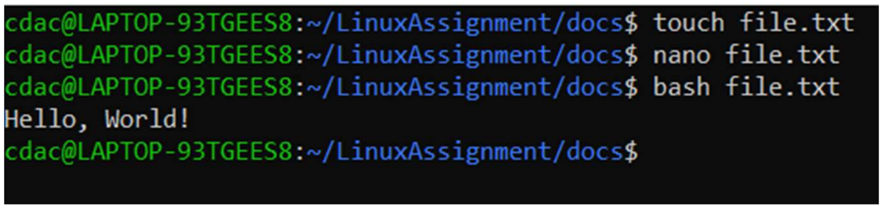
Part B

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

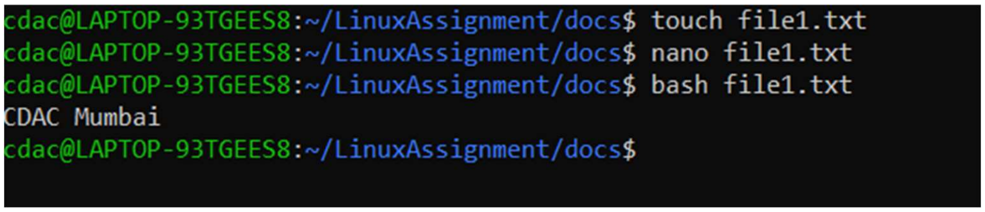
Identify the Incorrect Commands:
1chmodx is used to change file permissions. -incorrect
Chmod- Used to change file permissions
2. cpy is used to copy files and directories. Incorrect
cp is used to copy files and directories
3. mkfile is used to create a new file. Incorrect
Touch is used to create a new file
4. catx is used to concatenate files. Incorrect
Cat is used to concatenate files
5. rn is used to rename files . Incorrect
mv is used to rename files

Part C

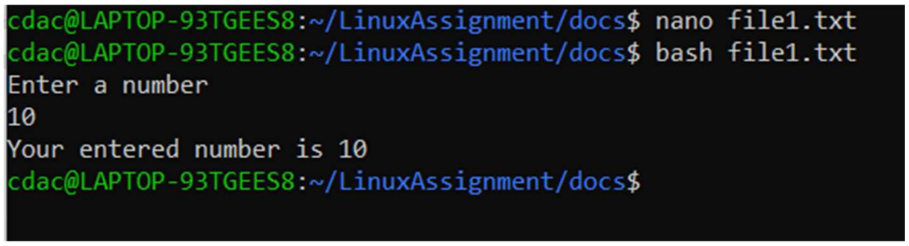
1) Write a shell script that prints "Hello, World!" to the terminal.
Ans:
```
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ touch file.txt
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ nano file.txt
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ bash file.txt
Hello, World!
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$
```


```
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ touch file.txt
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ nano file.txt
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ bash file.txt
Hello, World!
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$
```

2) Question 2: Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable.
   Ans:
   cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ touch file1.txt
   cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ nano file1.txt
   cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ bash file1.txt
   CDAC Mumbai
   cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$

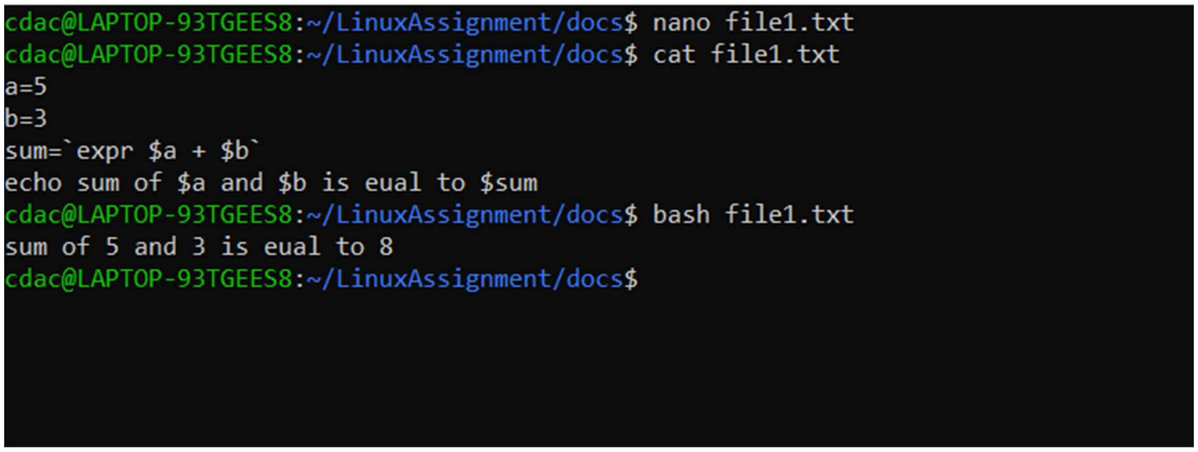

```
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ touch file1.txt
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ nano file1.txt
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ bash file1.txt
CDAC Mumbai
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$
```

3) Question 3: Write a shell script that takes a number as input from the user and prints it.
   Ans
   cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ nano file1.txt
   cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ bash file1.txt
   Enter a number
   10
   Your entered number is 10
   cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$


```
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ nano file1.txt
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ bash file1.txt
Enter a number
10
Your entered number is 10
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$
```

4) Question 4: Write a shell script that performs addition of two numbers (e.g., 5 and 3) and prints the result.
   cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ nano file1.txt
   cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ cat file1.txt
   a=5
   b=3

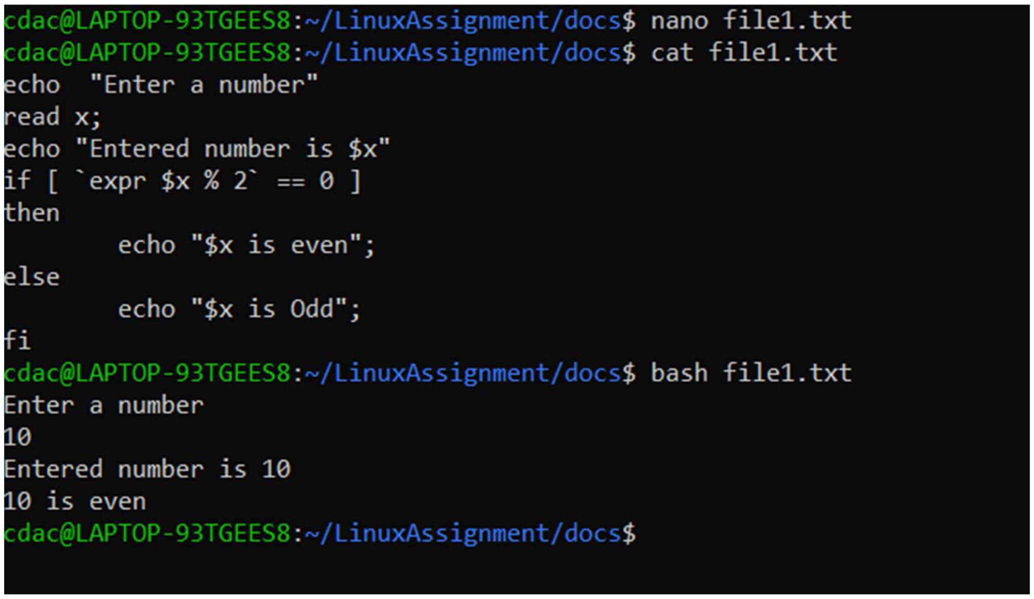
```
sum=`expr $a + $b`
echo sum of $a and $b is eual to $sum
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ bash file1.txt
sum of 5 and 3 is eual to 8
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$
```



```
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ nano file1.txt
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ cat file1.txt
a=5
b=3
sum=`expr $a + $b`
echo sum of $a and $b is eual to $sum
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ bash file1.txt
sum of 5 and 3 is eual to 8
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$
```

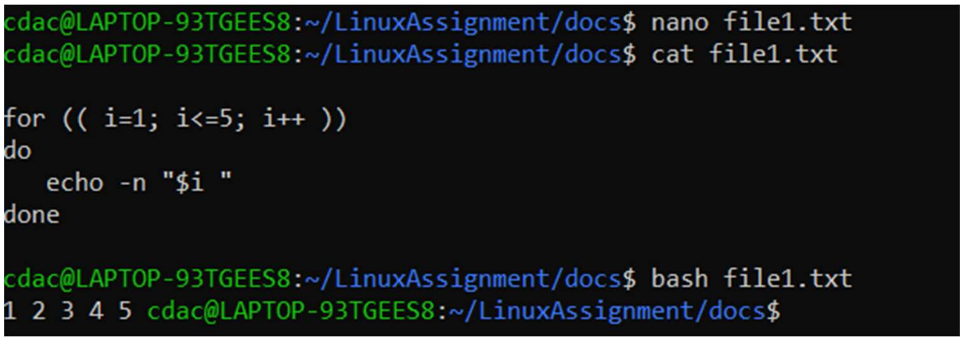
5) Question 5: Write a shell script that takes a number as input and prints "Even" if it is even, otherwise prints "Odd".

```
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ nano file1.txt
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ cat file1.txt
echo  "Enter a number"
read x;
echo "Entered number is $x"
if [ `expr $x % 2` == 0 ]
then
        echo "$x is even";
else
        echo "$x is Odd";
fi
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ bash file1.txt
Enter a number
10
Entered number is 10
10 is even
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$
```


```
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ nano file1.txt
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ cat file1.txt
echo  "Enter a number"
read x;
echo "Entered number is $x"
if [ `expr $x % 2` == 0 ]
then
        echo "$x is even";
else
        echo "$x is Odd";
fi
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ bash file1.txt
Enter a number
10
Entered number is 10
10 is even
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$
```

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

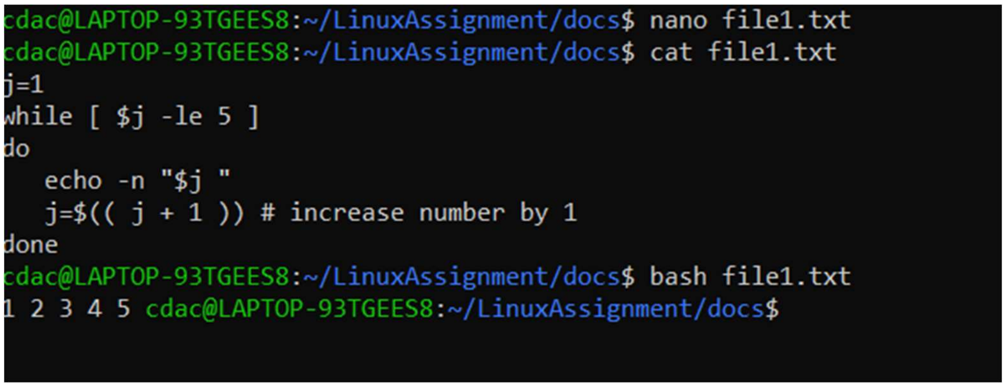
```
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ nano file1.txt
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ cat file1.txt

for (( i=1; i<=5; i++ ))
do
  echo -n "$i "
done

cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ bash file1.txt
1 2 3 4 5 cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$
```


```
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ nano file1.txt
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ cat file1.txt

for (( i=1; i<=5; i++ ))
do
   echo -n "$i "
done

cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ bash file1.txt
1 2 3 4 5 cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$
```

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

```
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ nano file1.txt
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ cat file1.txt
j=1
while [ $j -le 5 ]
```

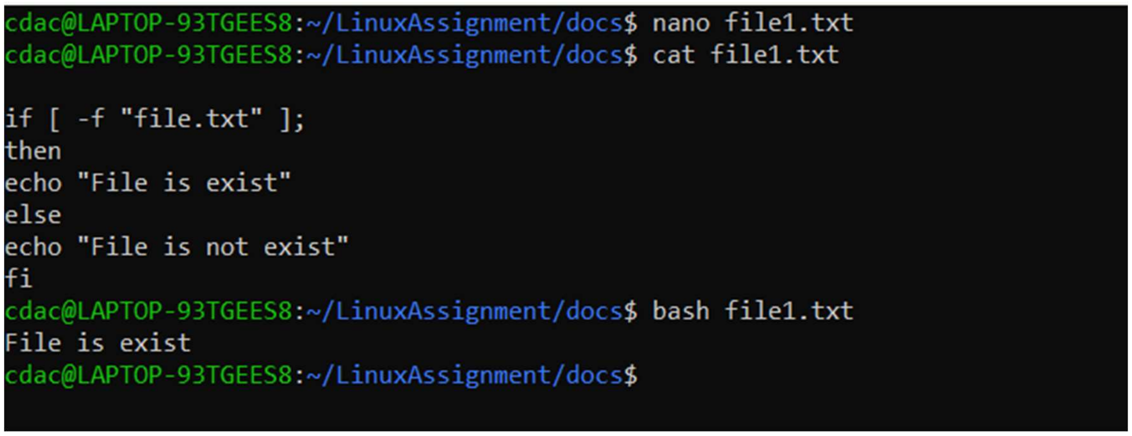
```
do
   echo -n "$j "
   j=$(( j + 1 )) # increase number by 1
done
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ bash file1.txt
1 2 3 4 5 cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$
```



```
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ nano file1.txt
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ cat file1.txt
j=1
while [ $j -le 5 ]
do
   echo -n "$j "
   j=$(( j + 1 )) # increase number by 1
done
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ bash file1.txt
1 2 3 4 5 cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$
```

8) 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".

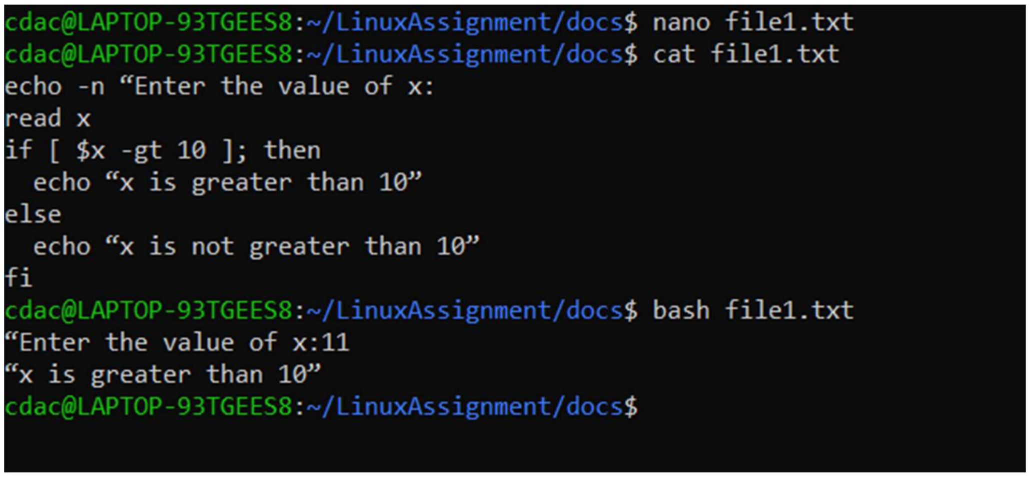
Ans:
```
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ nano file1.txt
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ cat file1.txt

if [ -f "file.txt" ];
then
echo "File is exist"
else
echo "File is not exist"
fi
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ bash file1.txt
File is exist
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$
```



```
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ nano file1.txt
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ cat file1.txt

if [ -f "file.txt" ];
then
echo "File is exist"
else
echo "File is not exist"
fi
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ bash file1.txt
File is exist
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$
```

9) 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.
   Ans:

```
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ nano file1.txt
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ cat file1.txt
echo -n “Enter the value of x:
read x
if [ $x -gt 10 ]; then
  echo “x is greater than 10”
else
  echo “x is not greater than 10”
fi
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ bash file1.txt
“Enter the value of x:11
“x is greater than 10”
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$
```

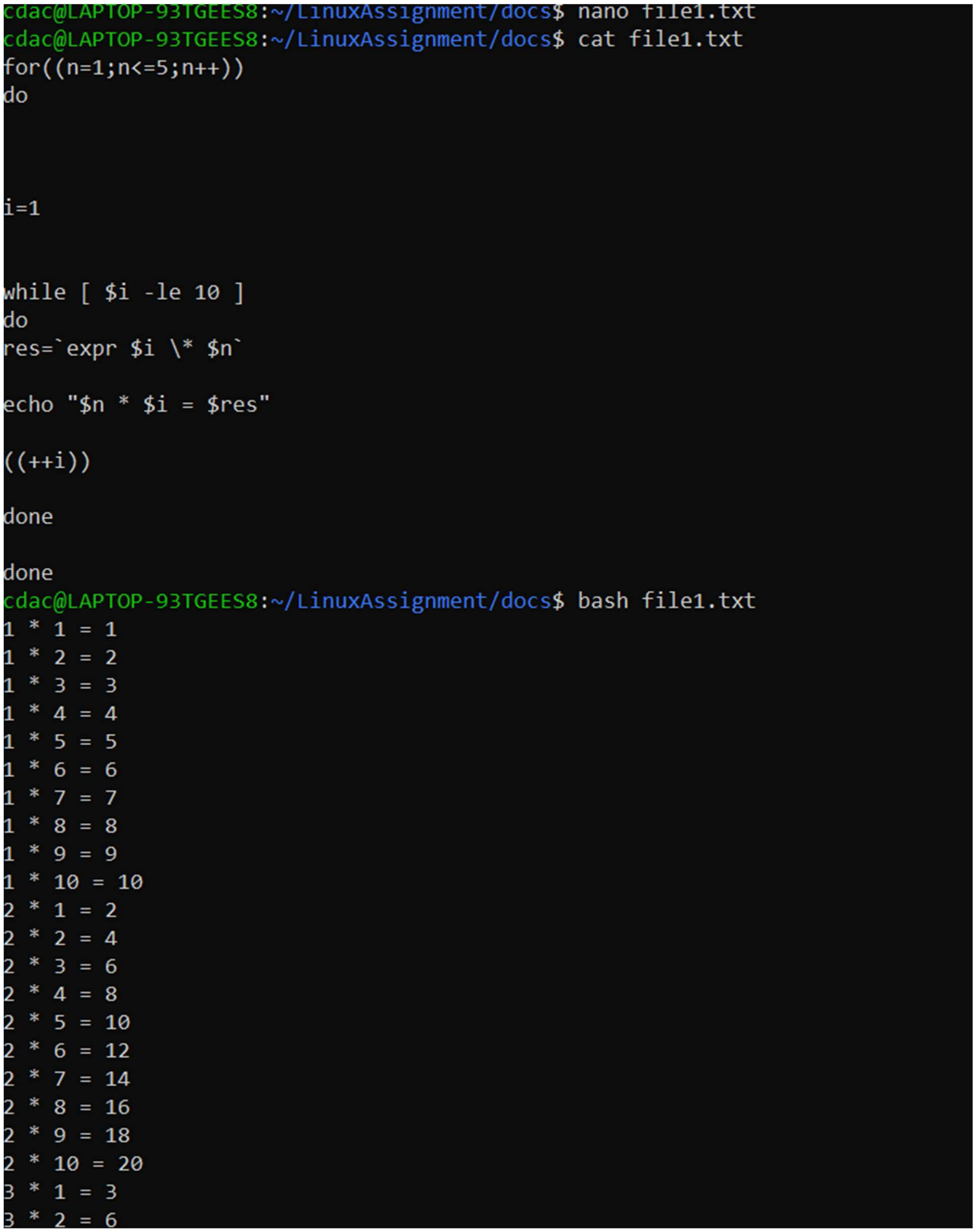


```
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ nano file1.txt
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ cat file1.txt
echo -n “Enter the value of x:
read x
if [ $x -gt 10 ]; then
  echo “x is greater than 10”
else
  echo “x is not greater than 10”
fi
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ bash file1.txt
“Enter the value of x:11
“x is greater than 10”
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$
```

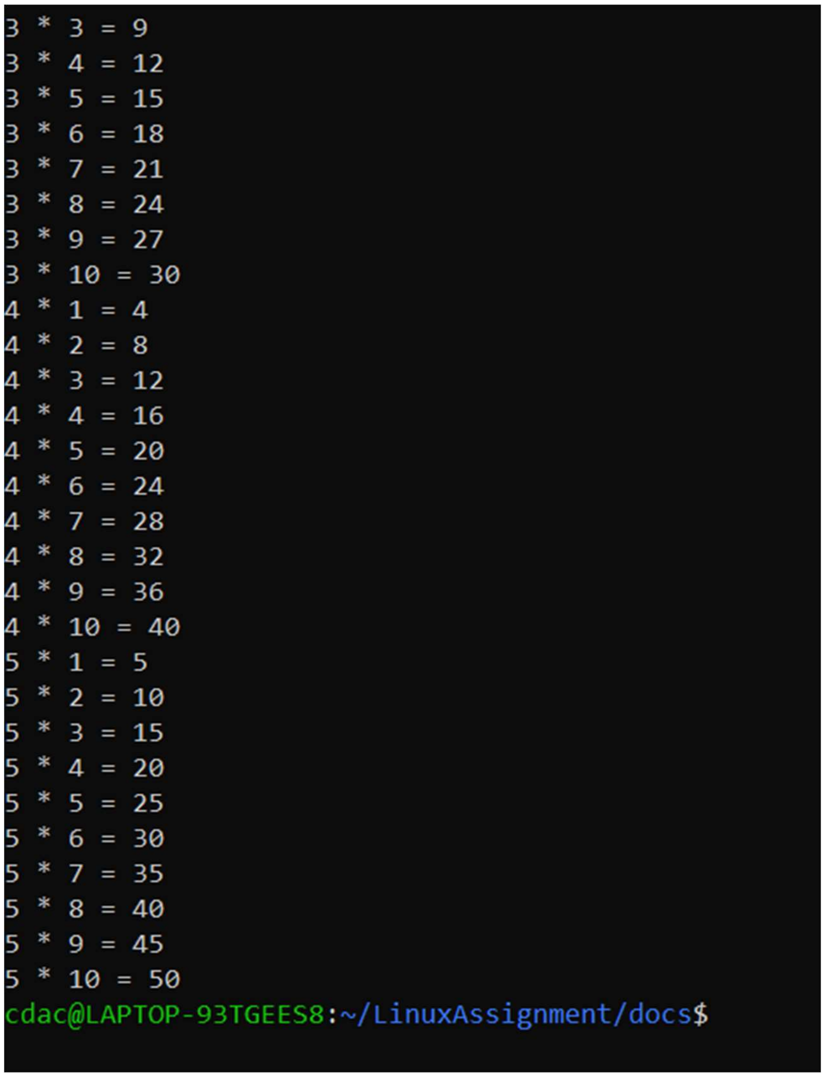
10) Question 10: Write a shell script that uses nested for loops to print a multiplication table for numbers from 1 to 5. The output should be formatted nicely, with each row representing a number and each column representing the multiplication result for that number.

Ans:
```
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ nano file1.txt
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ cat file1.txt
for((n=1;n<=5;n++))
do



i=1


while [ $i -le 10 ]
do
res=`expr $i \* $n`

echo "$n * $i = $res"

((++i))

done

done
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ bash file1.txt
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
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
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
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
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@LAPTOP-93TGEES8:~/LinuxAssignment/docs$
```



```
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ nano file1.txt
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ cat file1.txt
for((n=1;n<=5;n++))
do



i=1


while [ $i -le 10 ]
do
res=`expr $i \* $n`

echo "$n * $i = $res"

((++i))

done

done
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ bash file1.txt
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
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
3 * 1 = 3
3 * 2 = 6
```

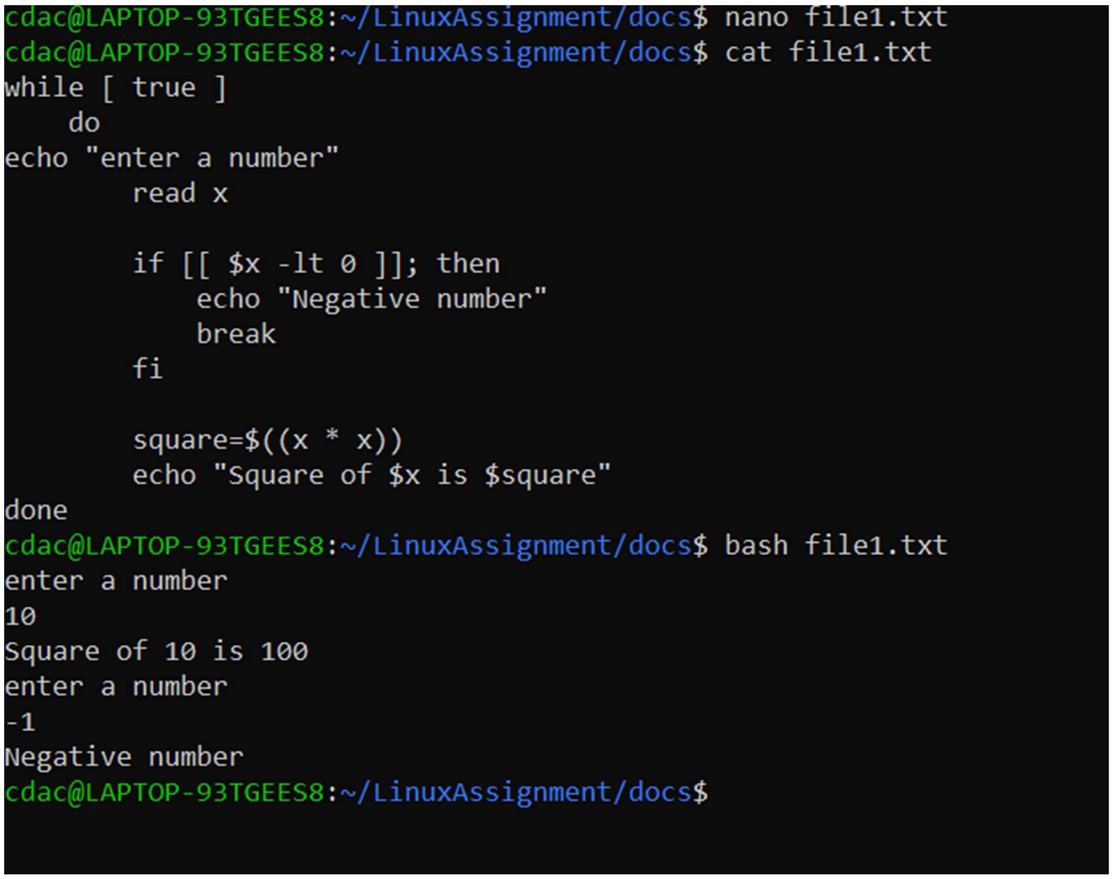

```
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30
4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
4 * 10 = 40
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$
```

11) 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.

Ans:

```
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ nano file1.txt
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ cat file1.txt
while [ true ]
   do
echo "enter a number"
```

```
        read x

        if [[ $x -lt 0 ]]; then
            echo "Negative number"
            break
        fi

        square=$((x * x))
        echo "Square of $x is $square"
done
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ bash file1.txt
enter a number
10
Square of 10 is 100
enter a number
-1
Negative number
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$
```



```
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ nano file1.txt
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ cat file1.txt
while [ true ]
    do
echo "enter a number"
        read x

        if [[ $x -lt 0 ]]; then
            echo "Negative number"
            break
        fi

        square=$((x * x))
        echo "Square of $x is $square"
done
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$ bash file1.txt
enter a number
10
Square of 10 is 100
enter a number
-1
Negative number
cdac@LAPTOP-93TGEES8:~/LinuxAssignment/docs$
```

Part E

1) **First-Come, First-Served (FCFS) scheduling**

process	Arrival time	Burst Time	Waiting time
P1	0	5	0
P2	1	3	4
P3	2	6	6

Gantt chart

0	5	8	14
P1	P2	P3	

Average waiting time =(0+4+6)/3=10/3=3.33333

2) **Shortest Job First (SJF) scheduling.**

Process	Arrival Time	Burst Time	Waiting time	TAT
P1	0	3	0	3
P2	1	5	7	12
P3	2	1	1	2
P4	3	4	1	5

Gantt chart

0	3	4	8	13
P1	P3	P4	P2	

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

3) **Priority Scheduling**

Note :1) Non preemptive

2)lower number represents higher priority

Process	Arrival time	Burst time	Priority	Waiting time
P1	0	6	3	0
P2	1	4	1	5
P3	2	7	4	10
P4	3	2	2	7

Gantt chart

0 6 10 12 19

P1	P2	P4	P3

Average waiting time =(0+5+10+7)/4=22/4=5.5

4) **Round Robin scheduling**
Quantum is 2 units

Process	Arrival time	Burst time	Waiting time	Turnaround time
P1	0	4	6	10
P2	1	5	8	13
P3	2	2	2	4
P4	3	3	7	10

Gantt chart

0 2 4 6 8 10 12 13 14

P1	P2	P3	P4	P1	P2	P4	P2

Average Turnaround Time (TAT) = (10 + 13 + 4 + 10) / 4 = 9.25

5) Consider a program that uses the fork() system call to create a child process. Initially, the parent process has a variable x with a value of 5. After forking, both the parent and child processes increment the value of x by 1. What will be the final values of x in the parent and child processes after the fork() call?
Ans: Before fork() call the parent process has a variable x initialized to 5.
After fork() call a new child process is created as a duplicate of the parent process, including its memory.
Both the parent and child have their own copies of x, which is still 5 at the moment of creation.
Incrementing x in both processes the parent process increments x by 1, making x = 6 in the parent.
The child process also increments its own x by 1, making x = 6 in the child.
Since the child and parent have separate memory spaces, changes in one do not reflect in the other.
In the parent process: x = 6 In the child process: x = 6