Assignment 2

Concepts of Operating System

Name: Mukta Wagh_KH

PartA

What will the following commands do?

1)echo "Hello, World!"

Ans:

It will print the given string i.e. "Hello, World!" On the terminal.

2) name="Productive"

Ans:

Name is the variable, productive is a value, and this is basically assigning value to variable.

3) touch file.txt

Ans:

It will create a new file name as file.txt.

4) ls -a

Ans:

Lists all files including hidden files.

5) rm file.txt

Ans:

Remove the given file.

6) cp file1.txt file2.txt

Ans:

Copies file1.txt to file2.txt

7) mv file.txt /path/to/directory/

Ans:

Move the file file.txt to the given directory.

8) chmod 755 script.sh

Ans:

Change the mode meaning give or deny the permission of read write and execution to the owner, group and other users. 7 meaning give all permission to the owner like read write and execution ,5 meaning only permission of read and execution to the group and other users.

9) grep "pattern" file.txt

Ans:

Search for the word "pattern" in file.txt.

10) kill PID

Ans:

Used to terminate the process using process ID.

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

Ans:

Here we use shell meta character && and it verify that each step is executes then and only if the previous one succeeds.

Mkdir mydir: it creates a directory name as mydir

Cd mydir: change directory to mydir

Touch file.txt: creates an empty file file.txt.

Echo hello world: print the given string

>: Redirection, It allow you to use output of one command as an input of other command

Cat file.txt: displays the content of file.txt.

Overall, it creates a directory, move it, creates a file, writes text into it, and then displays its contents.

12) ls -l | grep ".txt"

Ans:

Ls –l: lists the file and directory in details in current directory.

13) cat file1.txt file2.txt | sort | uniq

Ans:

Cat: combine these two files

| sort | uniq: pipping sort these files meaning sort the combine content and remove duplicates

14) ls -l | grep "^d"

Ans:

grep "^d": filters the output of ls-l and matches only lines where the first char is d.

15) grep -r "pattern" /path/to/directory/

Ans:

grep -r: The –r tells grep to search recursively.

16) cat file1.txt file2.txt | sort | uniq –d

Ans:

uniq –d: The –d with uniq shows only the lines that appear more than once in the sorted output.

17) chmod 644 file.txt

Ans:

It will change the permission of file.txt so that owner can read and write and group and others can read only.

18) cp -r source_directory destination_directory

Ans:

Copy the entire directory and its content into the given destination_directory.

19) find /path/to/search -name "*.txt"

Ans:

Find: it searches the file in the directory

Overall, it displays a list of all .txt files located within the specified path

20) chmod u+x file.txt

Ans:

u+x : u refer to user and x refer to execute so; the file owner gives permission on file.txt making it executable.

21) echo $PATH

Ans:

$PATH: The environment variable that contains a list of directories that are searched for executables files and displays it.

Part B

Identify True or False:

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

Ans: TRUE

2. mv is used to move files and directories.

Ans: TRUE

3. cd is used to copy files and directories.

Ans: FALSE

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

Ans: TRUE

5. grep is used to search for patterns in files.

Ans: TRUE

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

Ans: TRUE

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

Ans: TRUE

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

Ans: TRUE

Identify the Incorrect Commands:

1. chmodx is used to change file permissions.

2. cpy is used to copy files and directories.

3. mkfile is used to create a new file.

4. catx is used to concatenate files.

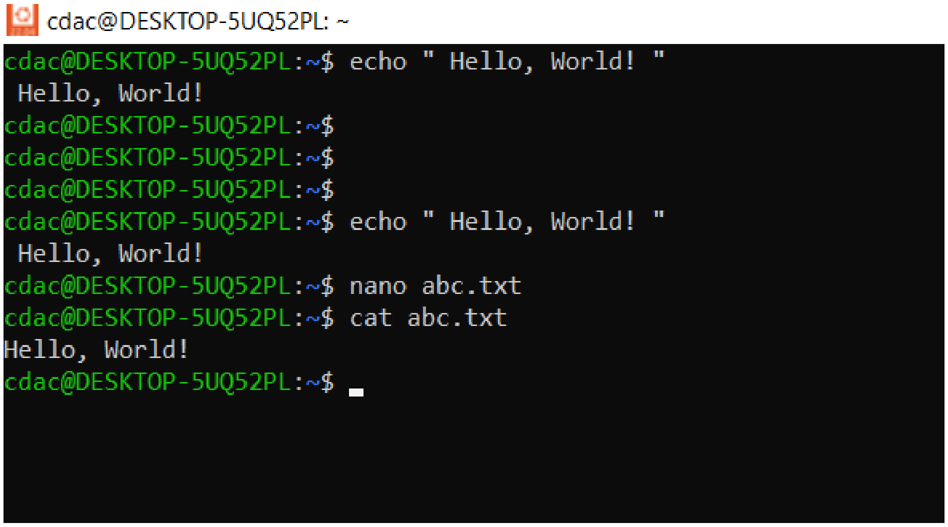
5. rn is used to rename files.

ANS: all are partially or completely incorrect

[Chmod not chmodx, cp not cp, touch not mkfile, cat not catx , mv not rn]

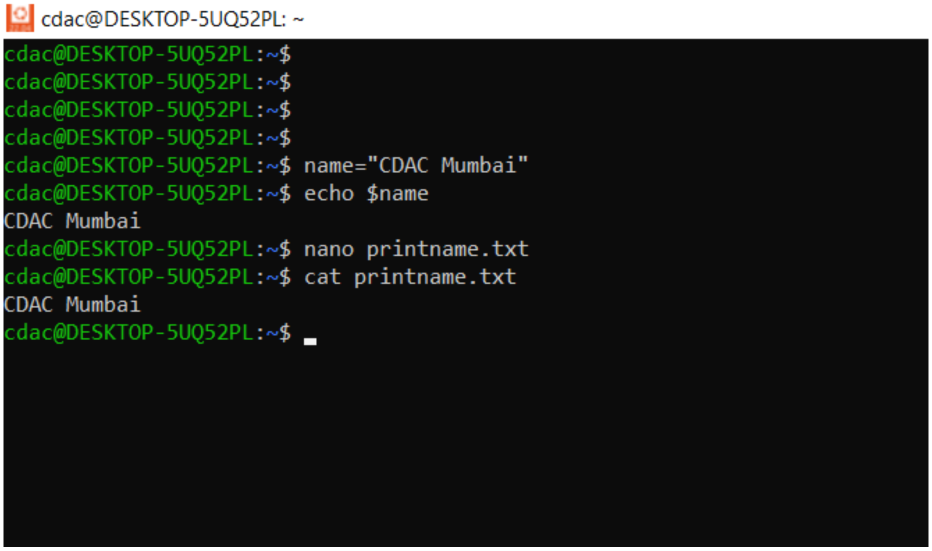
PartC

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



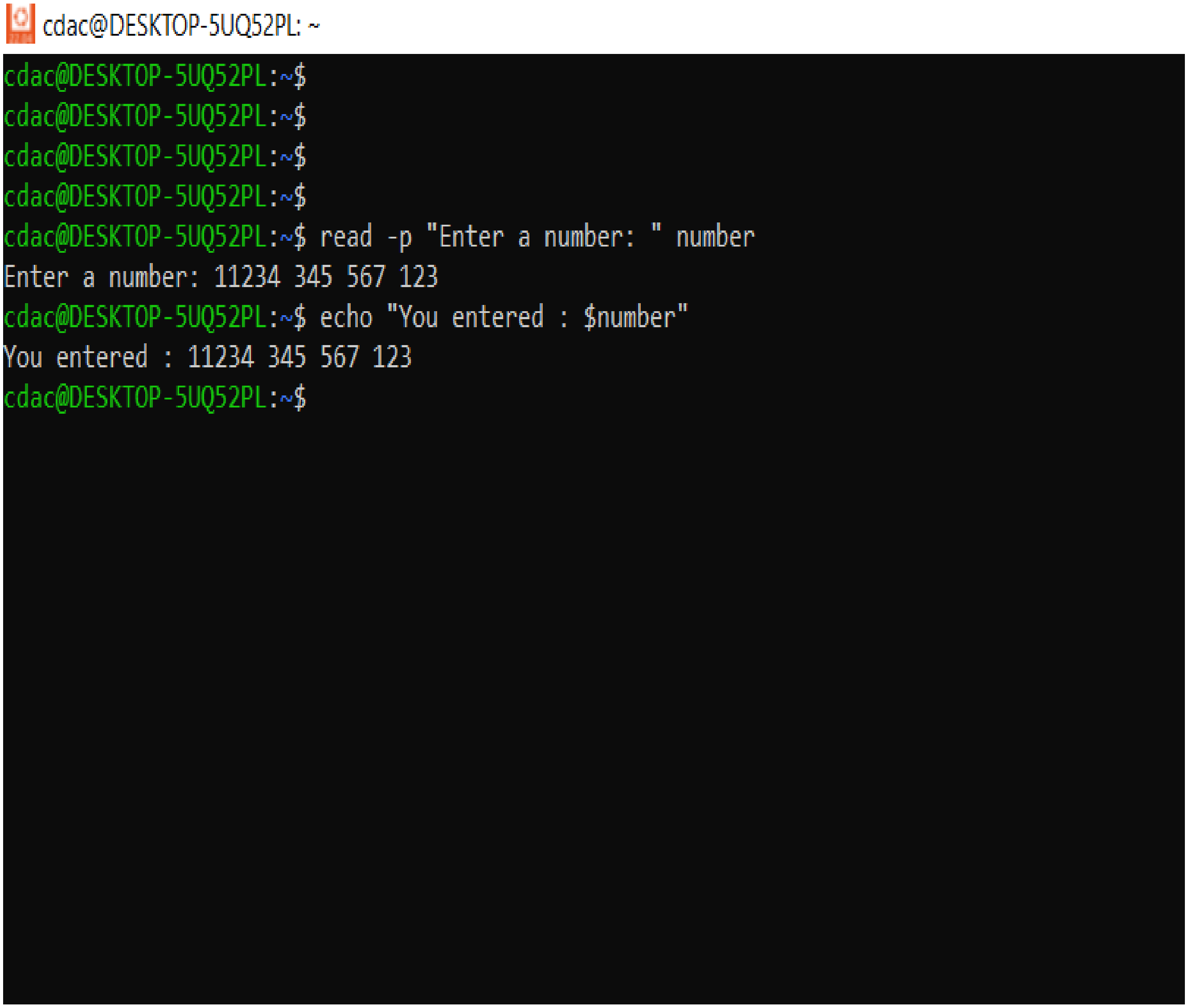
```
cdac@DESKTOP-5UQ52PL:~$ echo " Hello, World! "
 Hello, World!
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$ echo " Hello, World! "
 Hello, World!
cdac@DESKTOP-5UQ52PL:~$ nano abc.txt
cdac@DESKTOP-5UQ52PL:~$ cat abc.txt
Hello, World!
cdac@DESKTOP-5UQ52PL:~$ _
```

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



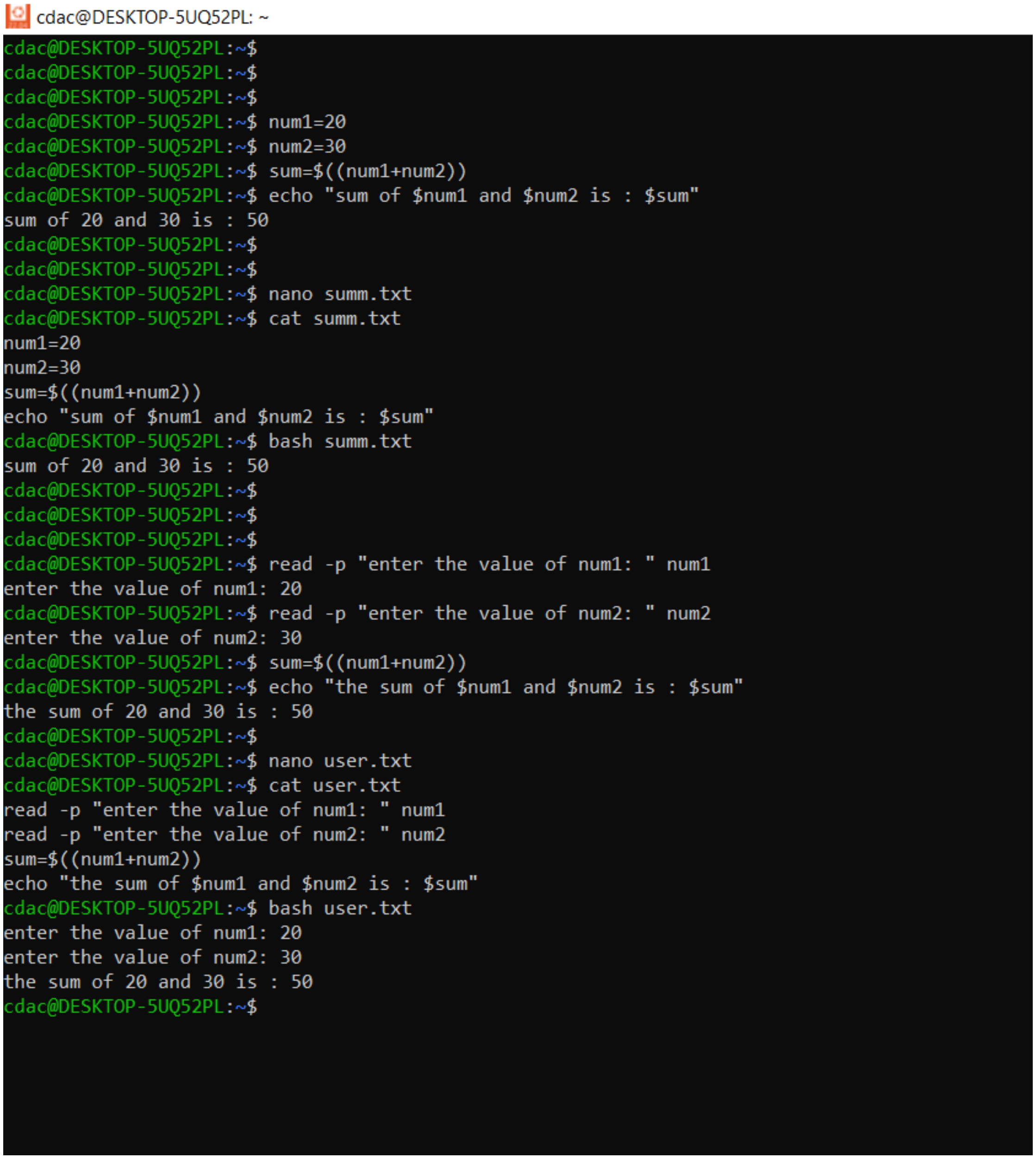
```
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$ name="CDAC Mumbai"
cdac@DESKTOP-5UQ52PL:~$ echo $name
CDAC Mumbai
cdac@DESKTOP-5UQ52PL:~$ nano printname.txt
cdac@DESKTOP-5UQ52PL:~$ cat printname.txt
CDAC Mumbai
cdac@DESKTOP-5UQ52PL:~$ _
```

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



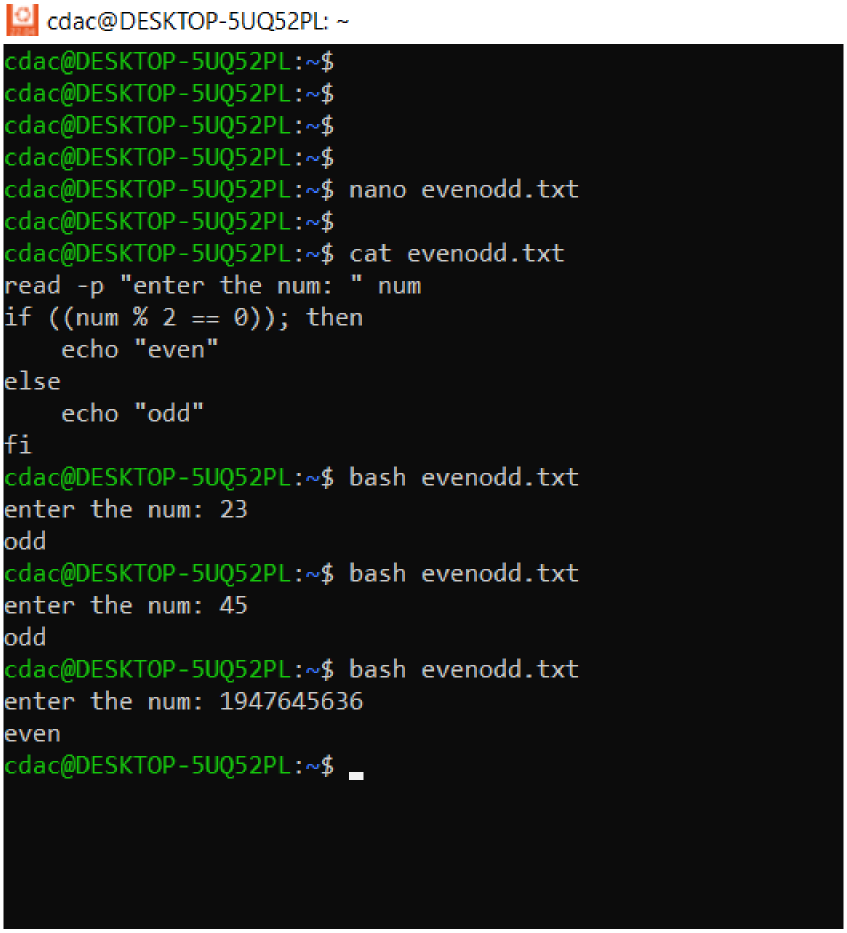
```
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$ read -p "Enter a number: " number
Enter a number: 11234 345 567 123
cdac@DESKTOP-5UQ52PL:~$ echo "You entered : $number"
You entered : 11234 345 567 123
cdac@DESKTOP-5UQ52PL:~$
```

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



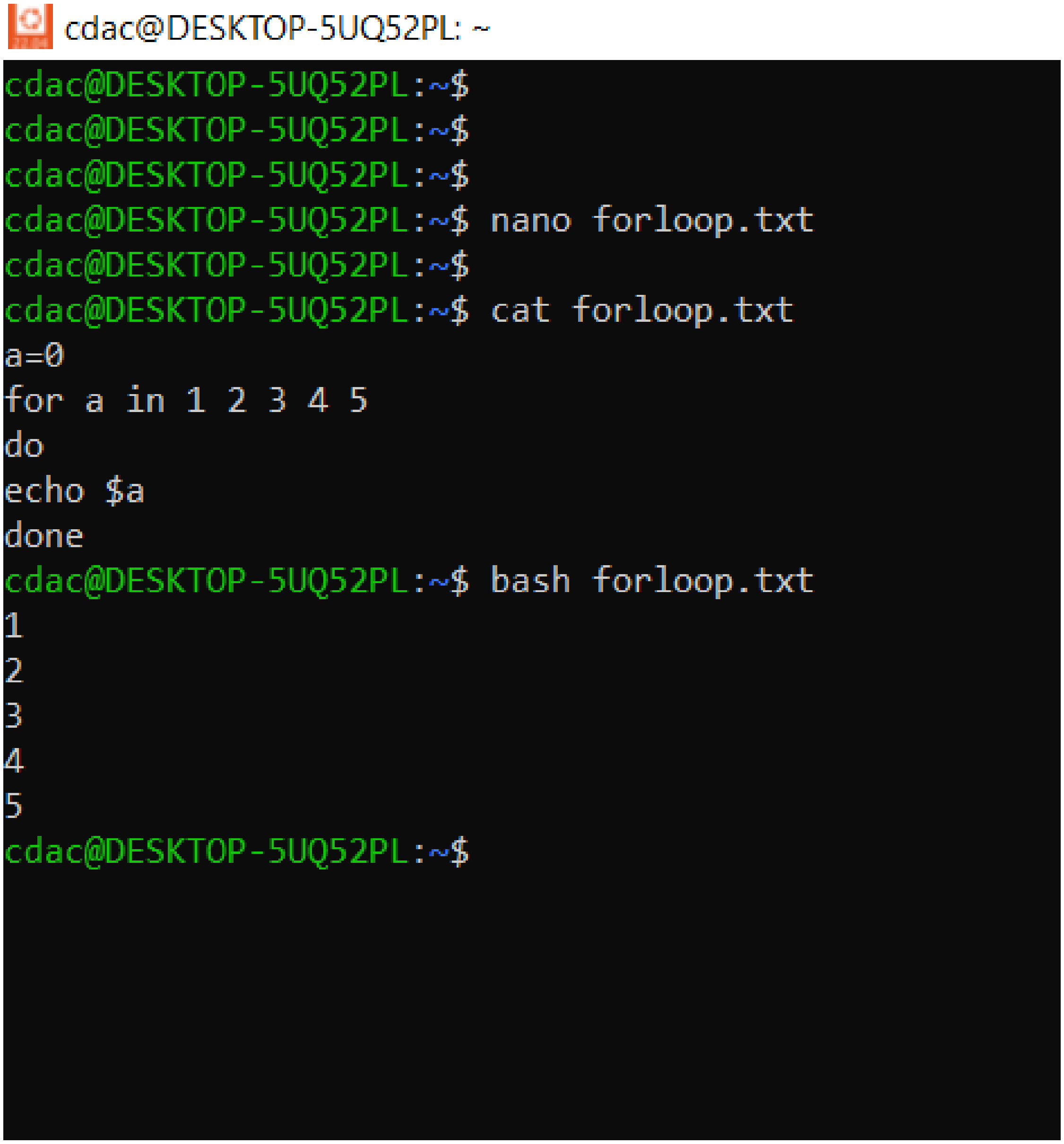
```
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$ num1=20
cdac@DESKTOP-5UQ52PL:~$ num2=30
cdac@DESKTOP-5UQ52PL:~$ sum=$((num1+num2))
cdac@DESKTOP-5UQ52PL:~$ echo "sum of $num1 and $num2 is : $sum"
sum of 20 and 30 is : 50
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$ nano summ.txt
cdac@DESKTOP-5UQ52PL:~$ cat summ.txt
num1=20
num2=30
sum=$((num1+num2))
echo "sum of $num1 and $num2 is : $sum"
cdac@DESKTOP-5UQ52PL:~$ bash summ.txt
sum of 20 and 30 is : 50
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$ read -p "enter the value of num1: " num1
enter the value of num1: 20
cdac@DESKTOP-5UQ52PL:~$ read -p "enter the value of num2: " num2
enter the value of num2: 30
cdac@DESKTOP-5UQ52PL:~$ sum=$((num1+num2))
cdac@DESKTOP-5UQ52PL:~$ echo "the sum of $num1 and $num2 is : $sum"
the sum of 20 and 30 is : 50
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$ nano user.txt
cdac@DESKTOP-5UQ52PL:~$ cat user.txt
read -p "enter the value of num1: " num1
read -p "enter the value of num2: " num2
sum=$((num1+num2))
echo "the sum of $num1 and $num2 is : $sum"
cdac@DESKTOP-5UQ52PL:~$ bash user.txt
enter the value of num1: 20
enter the value of num2: 30
the sum of 20 and 30 is : 50
cdac@DESKTOP-5UQ52PL:~$
```

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



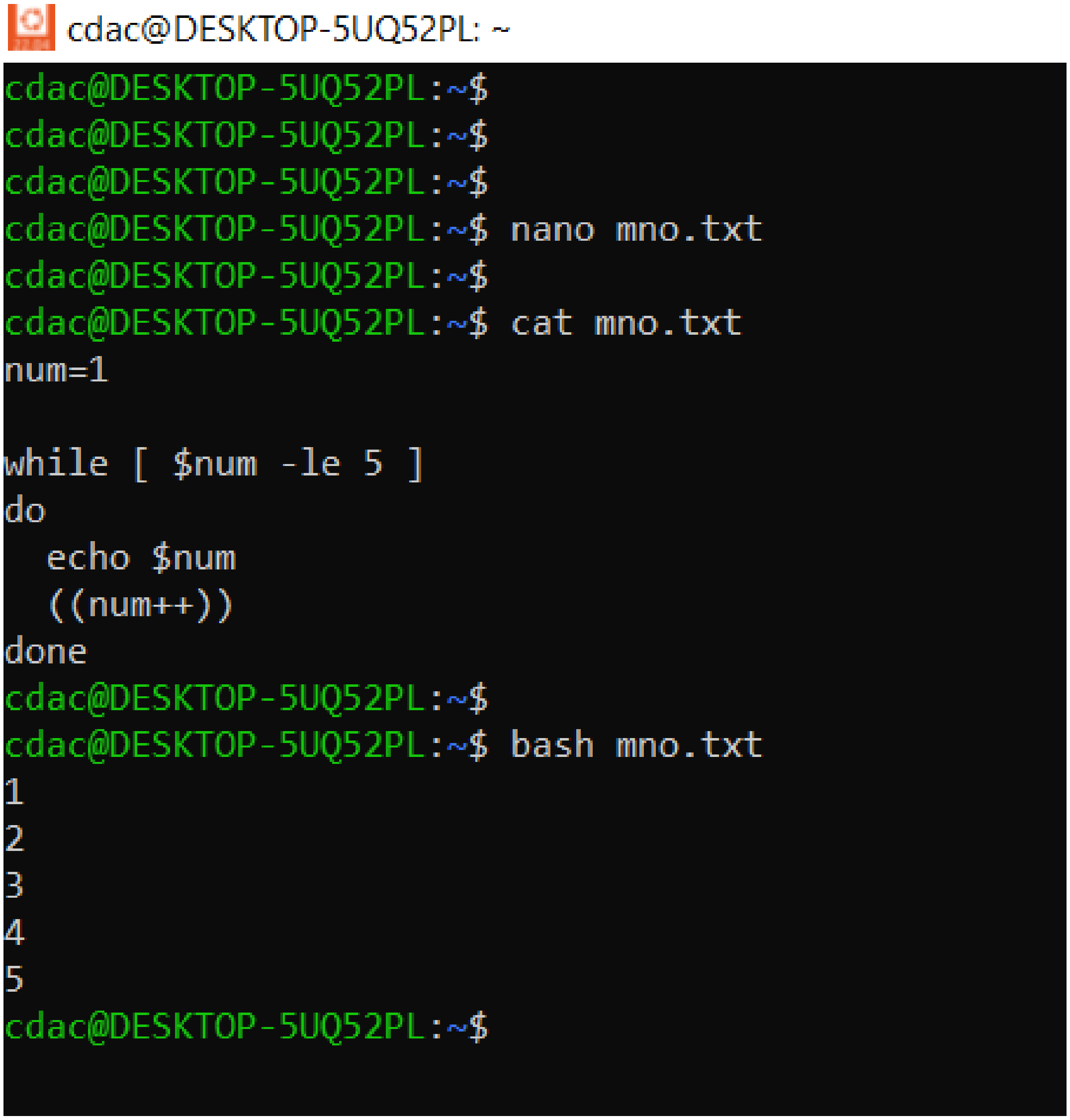
```
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$ nano evenodd.txt
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$ cat evenodd.txt
read -p "enter the num: " num
if ((num % 2 == 0)); then
    echo "even"
else
    echo "odd"
fi
cdac@DESKTOP-5UQ52PL:~$ bash evenodd.txt
enter the num: 23
odd
cdac@DESKTOP-5UQ52PL:~$ bash evenodd.txt
enter the num: 45
odd
cdac@DESKTOP-5UQ52PL:~$ bash evenodd.txt
enter the num: 1947645636
even
cdac@DESKTOP-5UQ52PL:~$
```

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



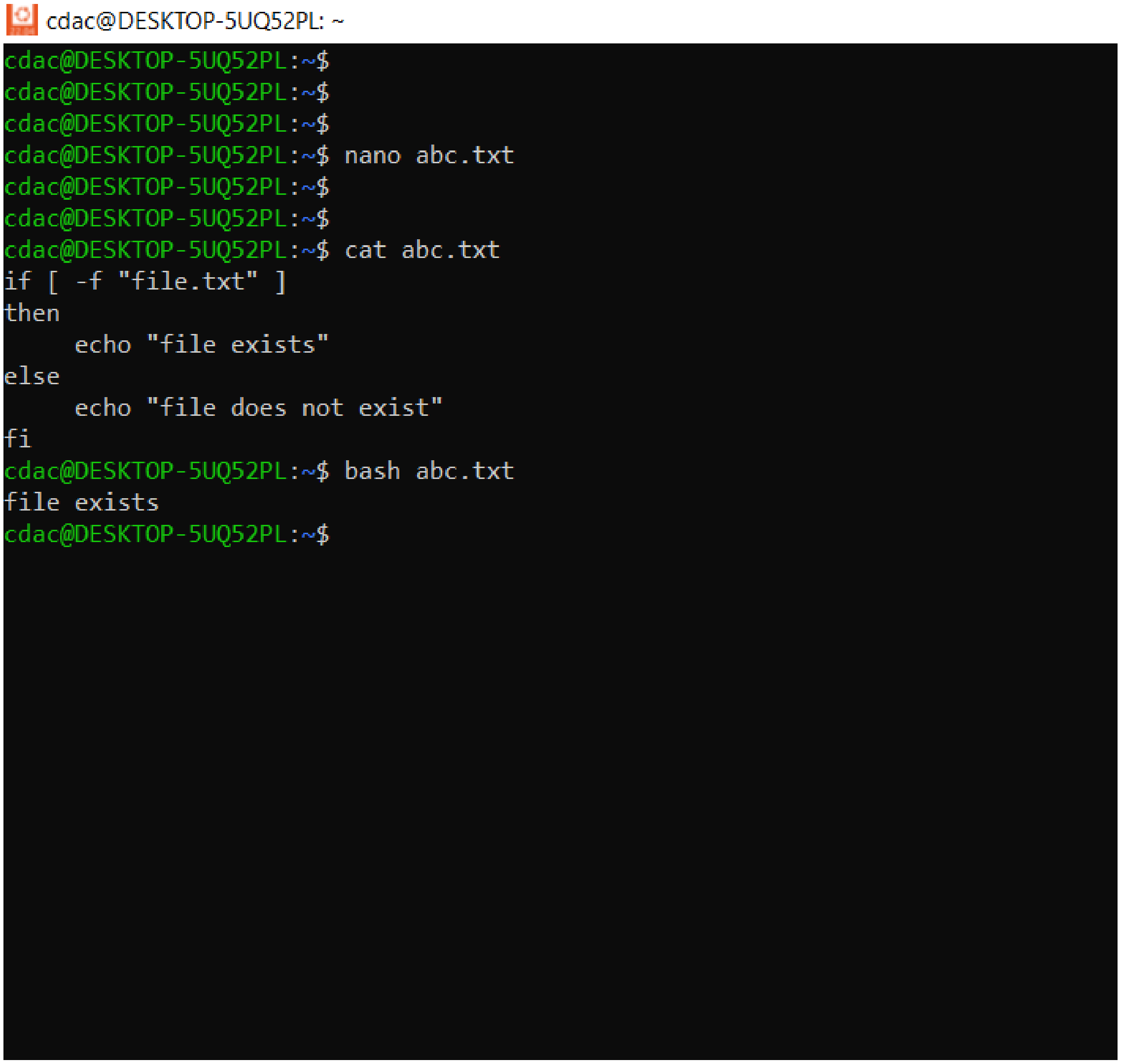
```
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$ nano forloop.txt
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$ cat forloop.txt
a=0
for a in 1 2 3 4 5
do
echo $a
done
cdac@DESKTOP-5UQ52PL:~$ bash forloop.txt
1
2
3
4
5
cdac@DESKTOP-5UQ52PL:~$
```

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



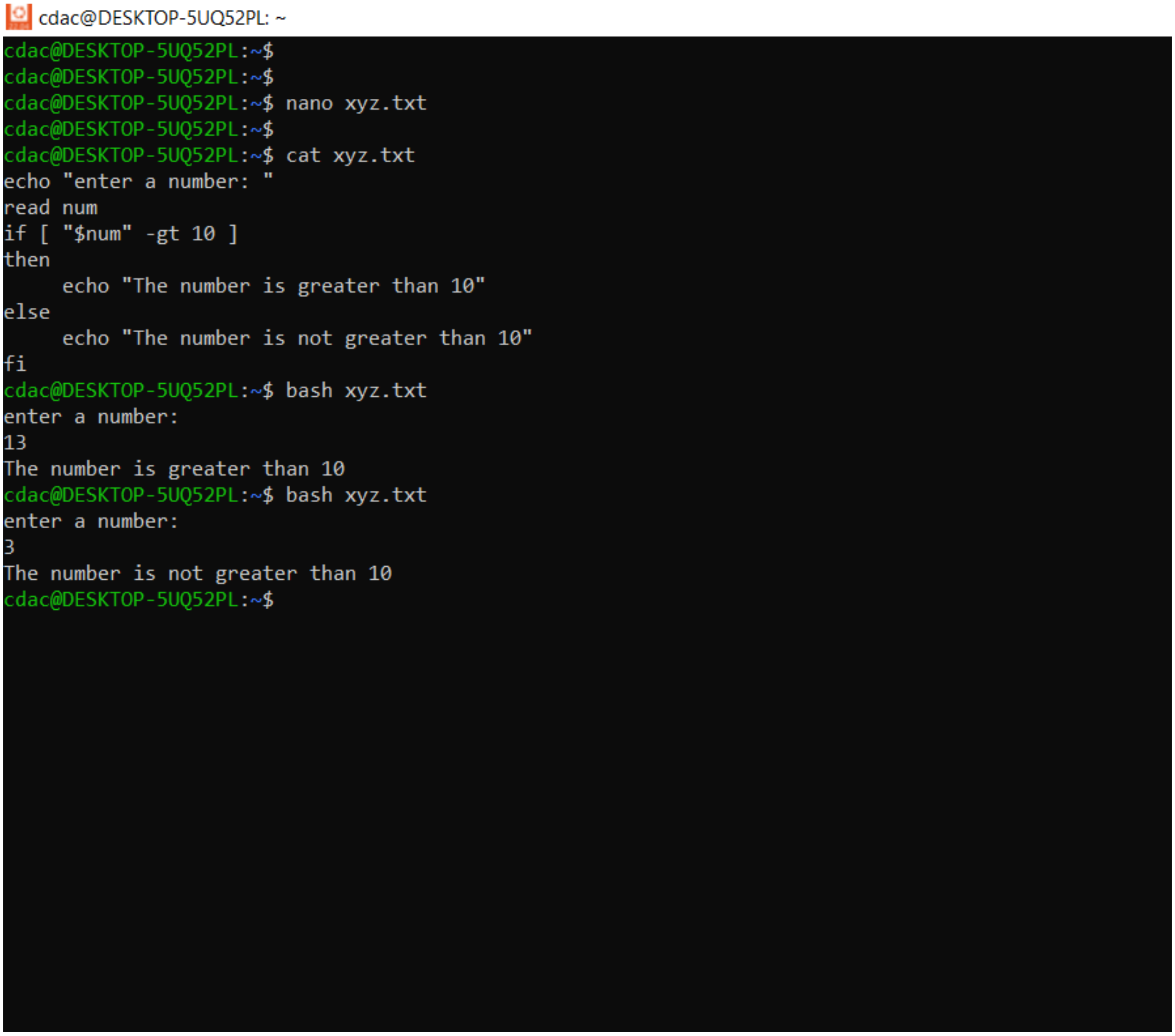
```
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$ nano mno.txt
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$ cat mno.txt
num=1

while [ $num -le 5 ]
do
  echo $num
  ((num++))
done
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$ bash mno.txt
1
2
3
4
5
cdac@DESKTOP-5UQ52PL:~$
```

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".



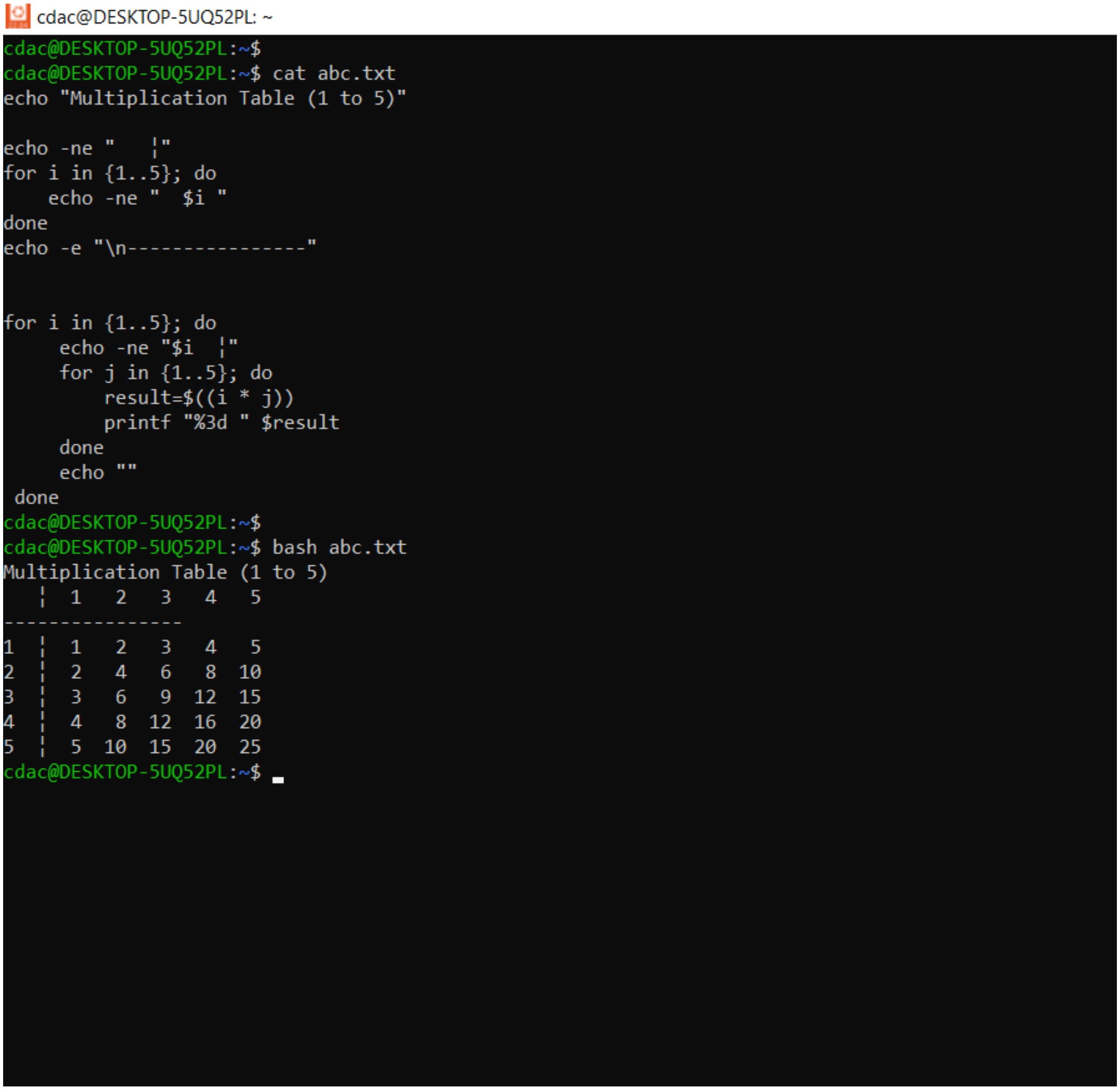
```
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$ nano abc.txt
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$ cat abc.txt
if [ -f "file.txt" ]
then
     echo "file exists"
else
     echo "file does not exist"
fi
cdac@DESKTOP-5UQ52PL:~$ bash abc.txt
file exists
cdac@DESKTOP-5UQ52PL:~$
```

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.



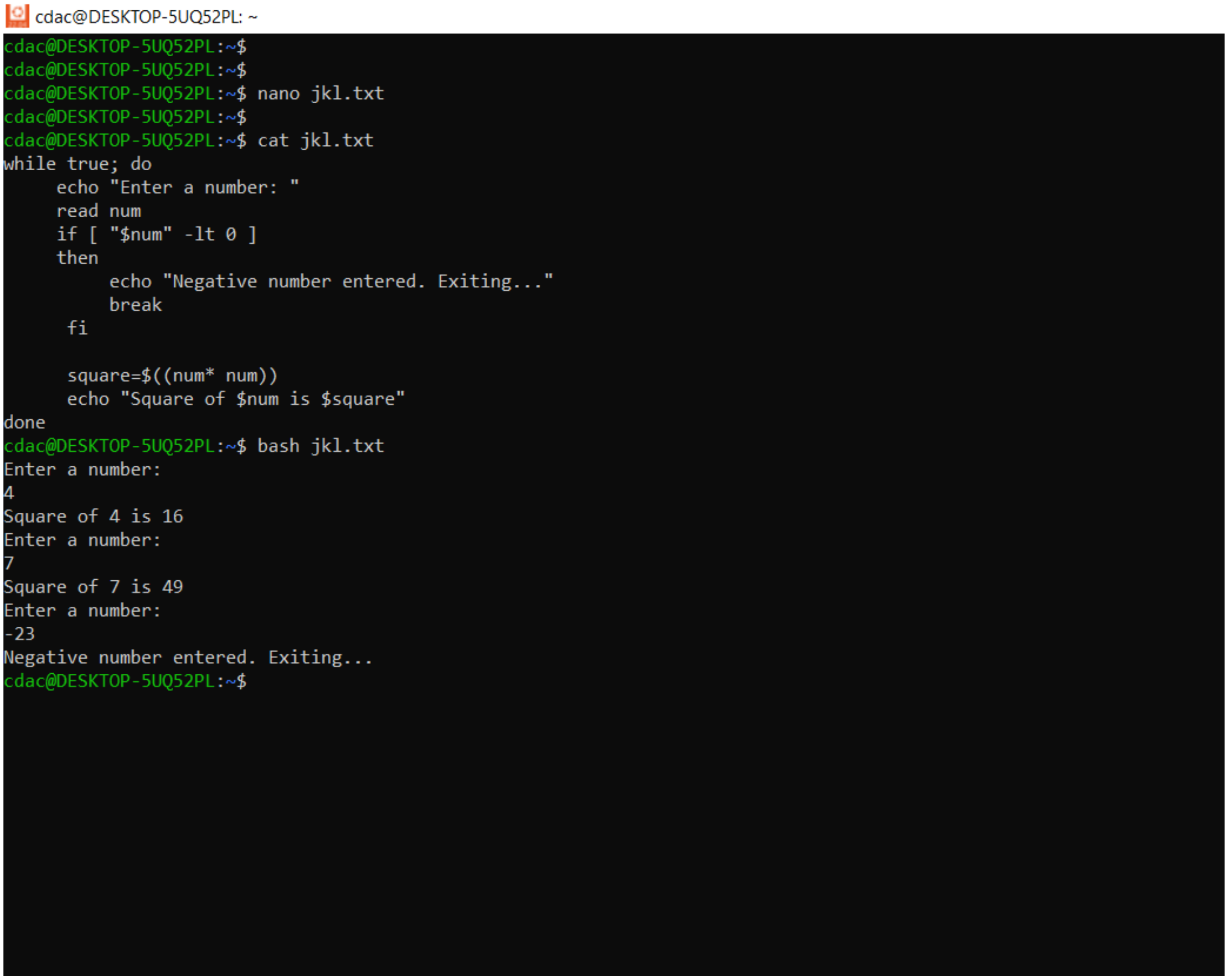
```
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$ nano xyz.txt
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$ cat xyz.txt
echo "enter a number: "
read num
if [ "$num" -gt 10 ]
then
     echo "The number is greater than 10"
else
     echo "The number is not greater than 10"
fi
cdac@DESKTOP-5UQ52PL:~$ bash xyz.txt
enter a number:
13
The number is greater than 10
cdac@DESKTOP-5UQ52PL:~$ bash xyz.txt
enter a number:
3
The number is not greater than 10
cdac@DESKTOP-5UQ52PL:~$
```

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.



```
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$ cat abc.txt
echo "Multiplication Table (1 to 5)"

echo -ne "   ¦"
for i in {1..5}; do
    echo -ne "  $i "
done
echo -e "\n----------------"


for i in {1..5}; do
     echo -ne "$i  ¦"
     for j in {1..5}; do
         result=$((i * j))
         printf "%3d " $result
     done
     echo ""
 done
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$ bash abc.txt
Multiplication Table (1 to 5)
   ¦  1   2   3   4   5
----------------
1  ¦  1   2   3   4   5
2  ¦  2   4   6   8  10
3  ¦  3   6   9  12  15
4  ¦  4   8  12  16  20
5  ¦  5  10  15  20  25
cdac@DESKTOP-5UQ52PL:~$ █
```

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.



```
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$ nano jkl.txt
cdac@DESKTOP-5UQ52PL:~$
cdac@DESKTOP-5UQ52PL:~$ cat jkl.txt
while true; do
     echo "Enter a number: "
     read num
     if [ "$num" -lt 0 ]
     then
          echo "Negative number entered. Exiting..."
          break
      fi

      square=$((num* num))
      echo "Square of $num is $square"
done
cdac@DESKTOP-5UQ52PL:~$ bash jkl.txt
Enter a number:
4
Square of 4 is 16
Enter a number:
7
Square of 7 is 49
Enter a number:
-23
Negative number entered. Exiting...
cdac@DESKTOP-5UQ52PL:~$
```

PART E

AT- Arrival Time

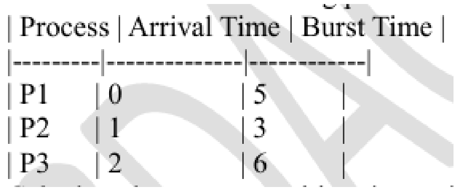
BT- Burst Time

RT- Response Time

WT- Waiting Time

TAT- Turn Arround Time

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



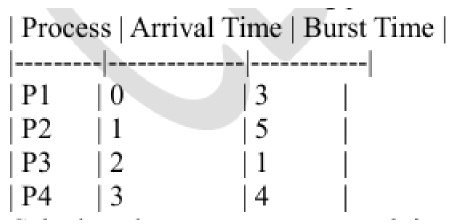
Process	Arrival Time	Burst Time
P1	0	5
P2	1	3
P3	2	6

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

ANS:

	Process	**AT**	**BT**	**RT**	**WT**
	P1	0	5	0	0
	P2	1	3	5	4
	P3	2	6	8	6
	Gantt Chart	**P1**	**P2**	**P3**	**.**
		0	5	8	14
	Average waiting time = 0+4+6/3				
	Average waiting time = 3.33 units				

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

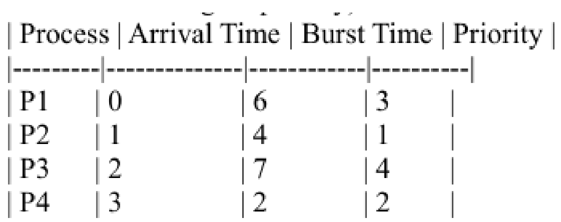
Calculate the average turnaround time using Shortest Job First (SJF) scheduling.

Ans:

	Process	**AT**	**BT**	**RT**	**WT**	**TAT**
	P1	0	3	0	0	3
	P2	1	5	8	7	12
	P3	2	1	3	1	2
	P4	3	4	4	1	5
	Gantt Chart	**P1**	**P3**	**P4**	**P2**	**.**
		0	3	4	8	13

Average TAT: 22/4= 5.5 units

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.

ANS:

WT1 and Gantt chart 1: for Non preemptive

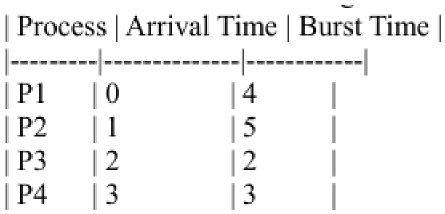
WT2 and Gantt chart 2 : for preemptive

	Priority	**Process**	**AT**	**BT**	**WT1**	**WT2**		
	3	P1	0	6	0	6		
	1	P2	1	4	5	0		
	4	P3	2	7	10	2		
	2	P4	3	2	7	10		
		Gantt Chart 1	**P1**	**P2**	**P4**	**P3**	**.**	
			0	6	10	12	19	
		Gantt Chart 2	**P1**	**P2**	**P4**	**P12**	**P3**	**.**
			0	1	5	7	12	19

Average waiting time1: 5.5

Average waiting time2: 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.

ANS:

	Process	**AT**	**BT**	**RT**	**WT**	**TAT**				
	P1	0	4	6	6	10				
	P2	1	5	8	8	13				
	P3	2	2	2	2	4				
	P4	3	3	7	7	10				
	Gantt Chart	**P1**	**P2**	**P3**	**P4**	**P12**	**P22**	**P42**	**P23**	**.**
		0	2	4	6	8	10	12	13	14
	Quantum =2 units									

Average TAT: 37/4 = 9.25 units

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:

When the fork () system call is used, it creates a child process that has its own copy of the parent's memory.

Before forking, the parent has a variable x = 5.

After the fork, both the parent and child have separate copies of x, still equal to 5.

Each process then increments x by 1, so both the parent and child have x = 6, but in their own separate memory.

In parent process, x=6.

In child process, x=6.