Assignment 2 - Part A

What will the following commands do?

1) echo "Hello, World!" :

echo command is use for printing the Message.this command print the Hello,World!. Even if quotes not present then also it print.

2) name="Productive :

This command is use for assign a variable. In the string we give the value, message so store in that.

3)touch file.txt :

touch command is use for create a new file. In that example file name is file.txt. In the linux shell command it is not compulsory to give a extension to the file.

4) ls -a :

ls command is use for Lists files and directories in the current directory. and **-a** option shows all files,including hidden ones.

5) rm file.txt :

rs command is use for remove or delete a file or directory. In that example file.txt is delete.

6) cp file1.txt file2.txt:

cp command is use for copy the file or directory. In that example file1.txt is copy and its content copy and paste into new file file2.txt.

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

mv command is use for rename or move the file or directory. In that example file.txt is move a file to the specific directory.

8) **chmod 755 script.sh :**

chmod command is use for give permission of read ,write,execute, permission to the owner,group, and other users to file script.sh.

a) 7 (Owner): Read (r), Write (w), Execute (x)

b) 5 (Group): Read (r), Execute (x)

c) 5 (Others) : Read (r), Execute (x)

8) grep "pattern" file.txt :

grep command is use for search specific pattern in a file. “pattern” is text or regular expression you want to find in file.txt.

10) kill PID :

kill command is use for terminate a process by its process id.

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

In that logical and is use (&&) we use this when we want more command in single line.

a)mkdir mydir:Creates a new directory named mydir.
b)cd mydir:Changes into one directory to another directory.in that example change to the mydir directory.
c)touch file.txt:Creates an empty file named file.txt.
d)echo "Hello, World!" > file.txt :Writes "Hello, World!" into file.txt.in that > is use for redirect menes one file's output copy into another file.in that message is copy to file.txt file.
e)cat file.txt :Displays the content of file.txt. what written in that file.

12) ls -l | grep ".txt" :

The above command uses piping to combine the output of both ls and grep command. ls -l is used to display the contents of current directory with details and grep ".txt" command is used to display all the files conating .txt pattern in their name.

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

The above command uses piping to combine the output of cat sort and uniq commands. First command i.e. **cat** command is used to display the contents of file1.txt followed by contents of file2.txt.
Sort: sort command is used to sort the file in alphabetically file1 and file2.txt are sorted separately in the result

uniq command is use to display only distinct lines in the result.duplicate line remove show only uniq line.

14) ls -l | grep "^d" :

ls command lists the files and directories in long format. grep "^d" command filters the output to show only lines that start with "d" which in the ls -l output indicates directories.

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

grep command is used to recursively search for given pattern “pattern” in the directory /path/to/directory, provided that such directory exists in first place. The output will the lines containing the “pattern” pattern in it

.

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

a) cat file1.txt file2.txt :display the contents of both files.
b) **sort** : Sorts the combined output of file1.txt and file2.txt by alphabumric way.
c)**uniq –d**: Filters and prints only duplicate lines.

17) chmod 644 file.txt :

The **chmod** command assigns read and write permissions to owner of the file file.txt and read permission to group users and other users respectively.

18) cp -r source_directory destination_directory:

The above command copies the source_directory to the destination_directory using the -r option, which ensures that all files and subdirectories inside source_directory are copied recursively.

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

find command is used for searching the files and directories. In that command searches /path/to/search directory and its subdirectories for any file ending with .txt pattern.

20) chmod u+x file.txt :

a) chmod: Command to change file permissions.
b) u:use to the user (owner) of the file.
c) +x : Adds execute permission.
d) file.txt : is the file name file.txt we perform this operation on that file.

21) echo $PATH :

This command displays the value of system environment variable that stores directories where executable programs are located.

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**, used to change the directory.

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. – **False**

Identify the Incorrect Commands:

1) chmodx is used to change file permissions.
Above command is incorrect chmod command is used to change file permissions.

2) cpy is used to copy files and directories.
Above command is incorrect cp command is used to copy files and directories.

3) mkfile is used to create a new file.
Above command is incorrect touch command is used to create a new file.

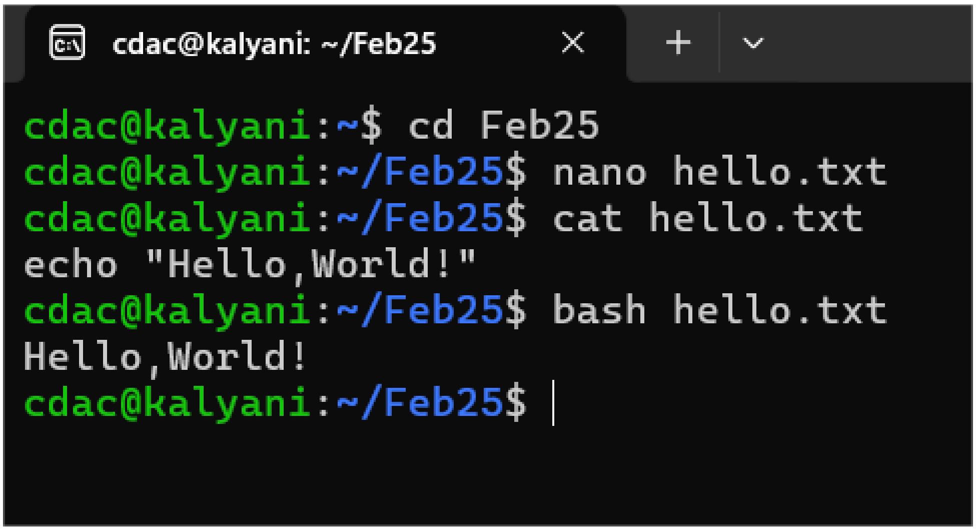
4) catx is used to concatenate files.
Above command is incorrect cat command is used to concatenate files.

5) rn is used to rename files.

Above command is incorrect mv command is used to rename.

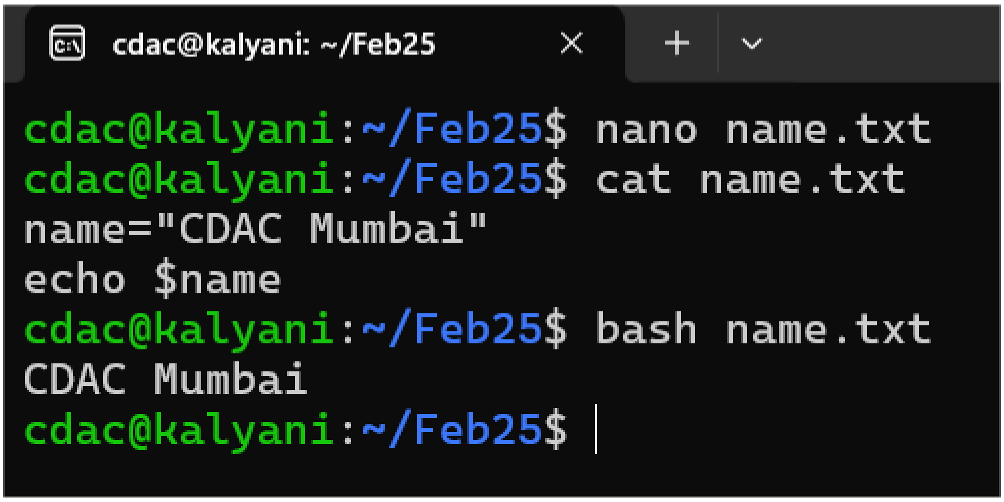
PART C

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



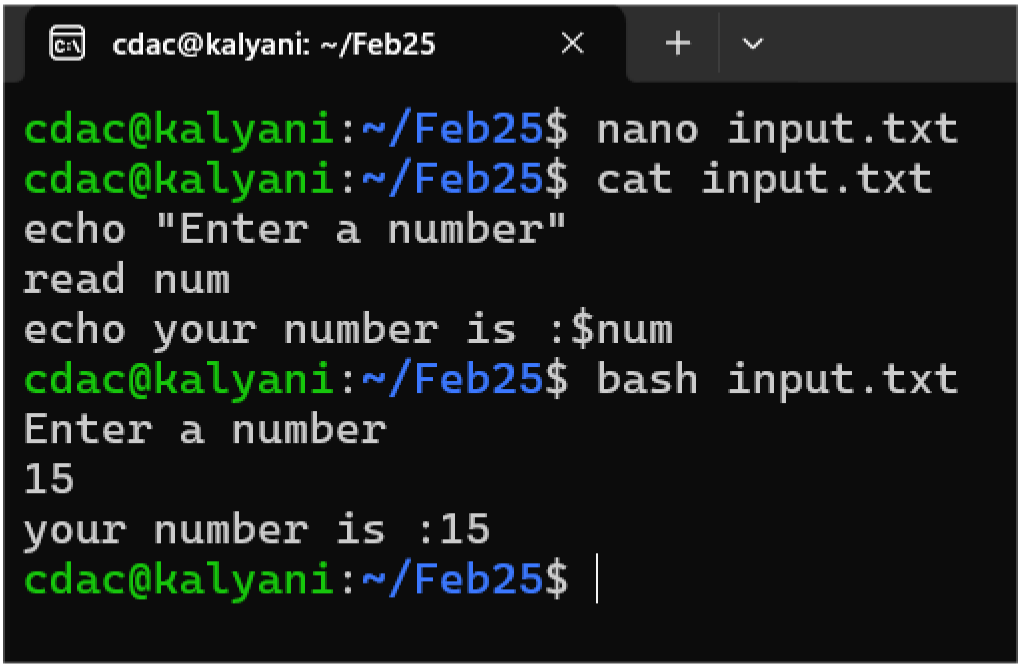
```
cdac@kalyani:~$ cd Feb25
cdac@kalyani:~/Feb25$ nano hello.txt
cdac@kalyani:~/Feb25$ cat hello.txt
echo "Hello,World!"
cdac@kalyani:~/Feb25$ bash hello.txt
Hello,World!
cdac@kalyani:~/Feb25$
```

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



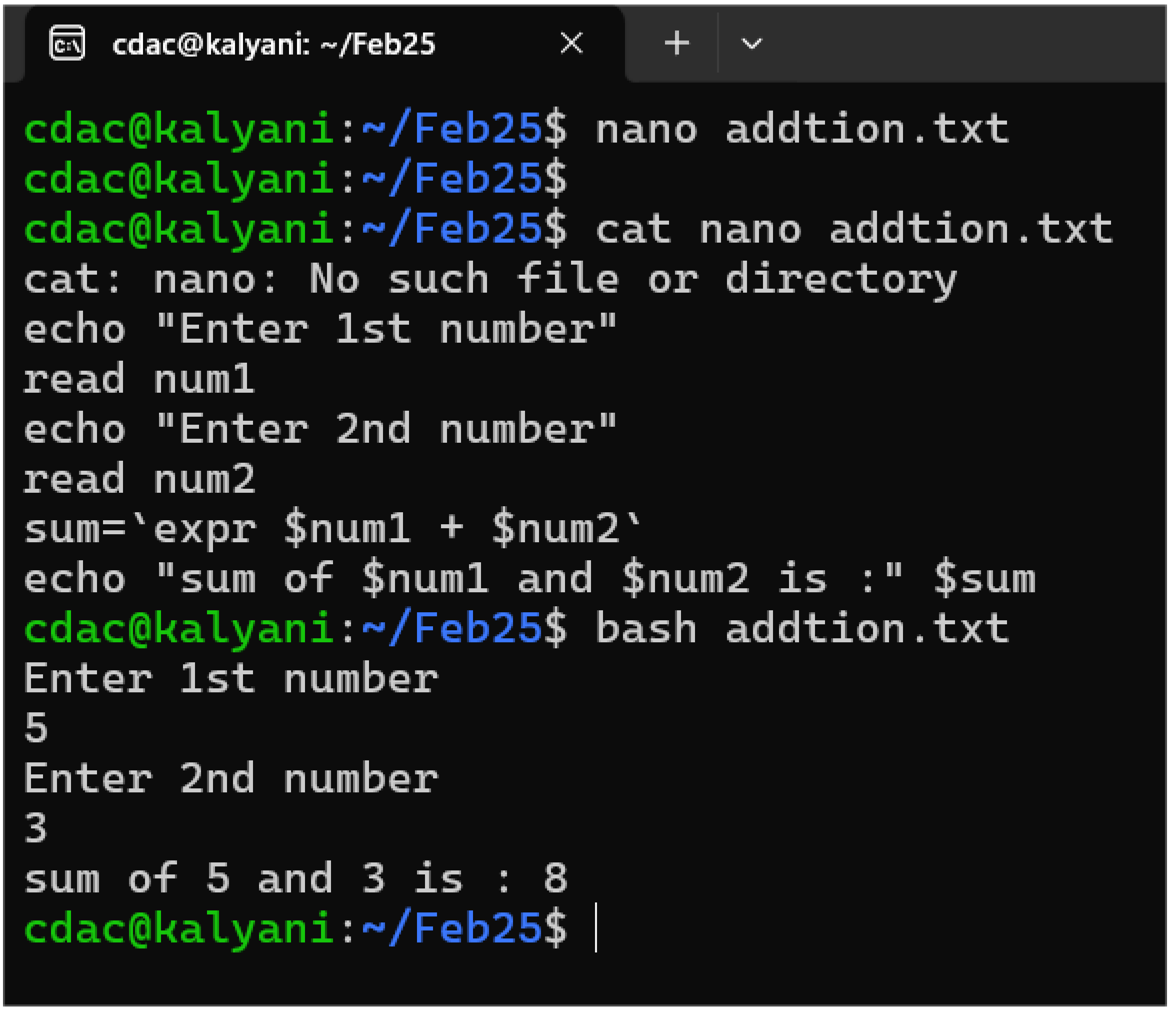
```
cdac@kalyani:~/Feb25$ nano name.txt
cdac@kalyani:~/Feb25$ cat name.txt
name="CDAC Mumbai"
echo $name
cdac@kalyani:~/Feb25$ bash name.txt
CDAC Mumbai
cdac@kalyani:~/Feb25$
```

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



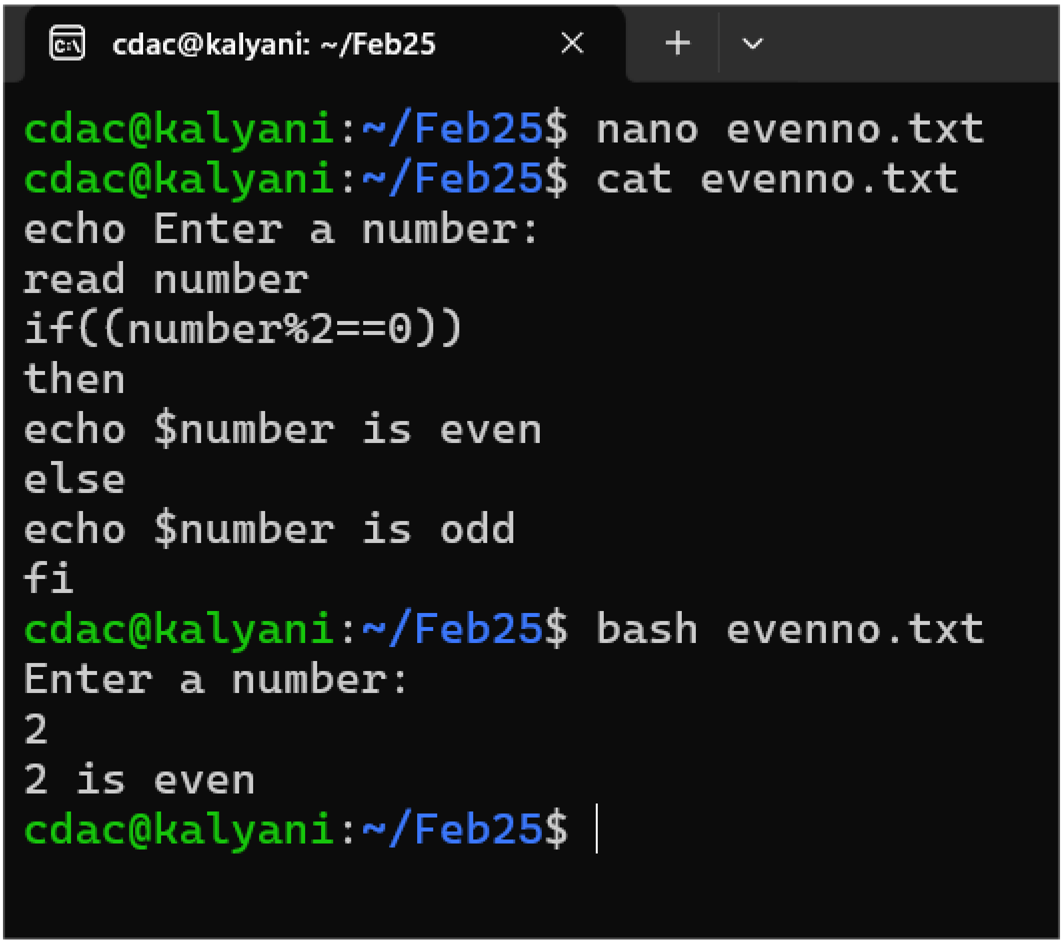
```
cdac@kalyani:~/Feb25$ nano input.txt
cdac@kalyani:~/Feb25$ cat input.txt
echo "Enter a number"
read num
echo your number is :$num
cdac@kalyani:~/Feb25$ bash input.txt
Enter a number
15
your number is :15
cdac@kalyani:~/Feb25$
```

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



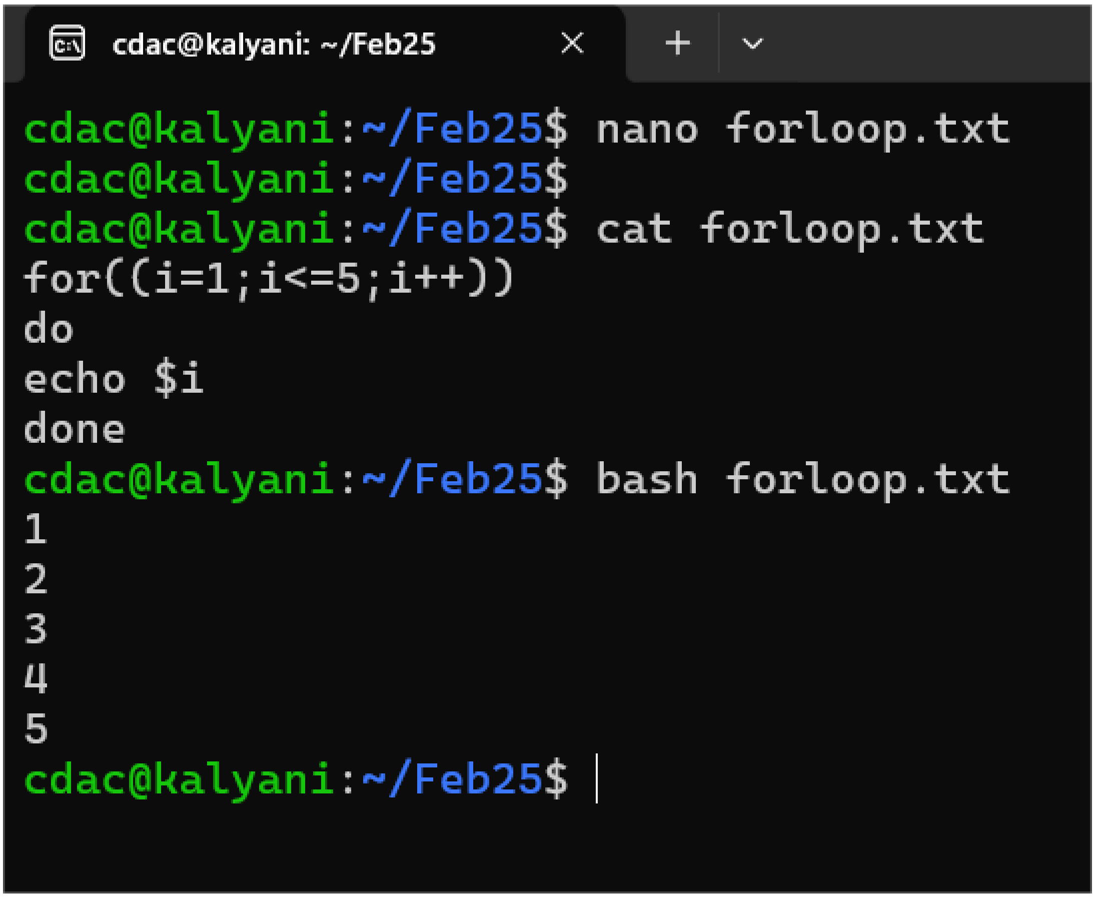
```
cdac@kalyani:~/Feb25$ nano addtion.txt
cdac@kalyani:~/Feb25$
cdac@kalyani:~/Feb25$ cat nano addtion.txt
cat: nano: No such file or directory
echo "Enter 1st number"
read num1
echo "Enter 2nd number"
read num2
sum=`expr $num1 + $num2`
echo "sum of $num1 and $num2 is :" $sum
cdac@kalyani:~/Feb25$ bash addtion.txt
Enter 1st number
5
Enter 2nd number
3
sum of 5 and 3 is : 8
cdac@kalyani:~/Feb25$
```

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



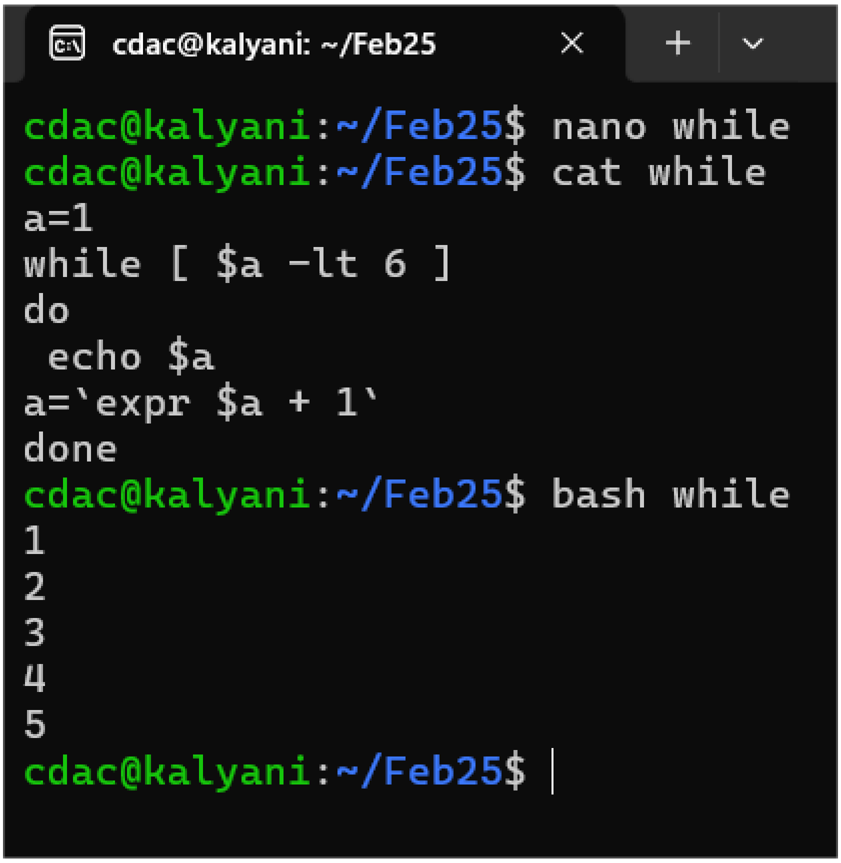
```
cdac@kalyani:~/Feb25$ nano evenno.txt
cdac@kalyani:~/Feb25$ cat evenno.txt
echo Enter a number:
read number
if((number%2==0))
then
echo $number is even
else
echo $number is odd
fi
cdac@kalyani:~/Feb25$ bash evenno.txt
Enter a number:
2
2 is even
cdac@kalyani:~/Feb25$ 
```

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



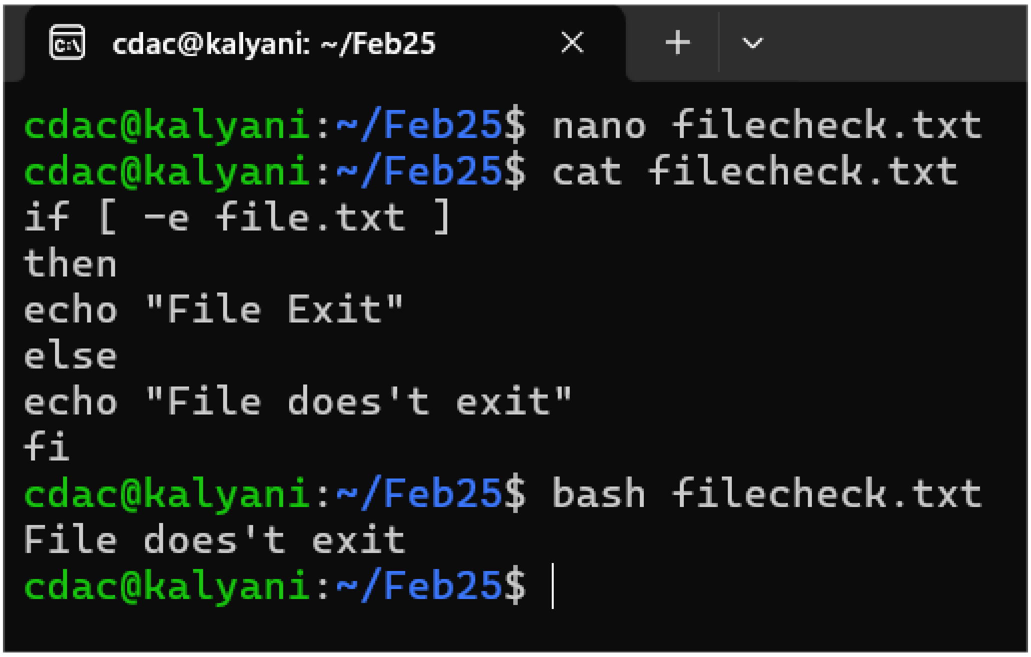
```
cdac@kalyani:~/Feb25$ nano forloop.txt
cdac@kalyani:~/Feb25$
cdac@kalyani:~/Feb25$ cat forloop.txt
for((i=1;i<=5;i++))
do
echo $i
done
cdac@kalyani:~/Feb25$ bash forloop.txt
1
2
3
4
5
cdac@kalyani:~/Feb25$
```

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



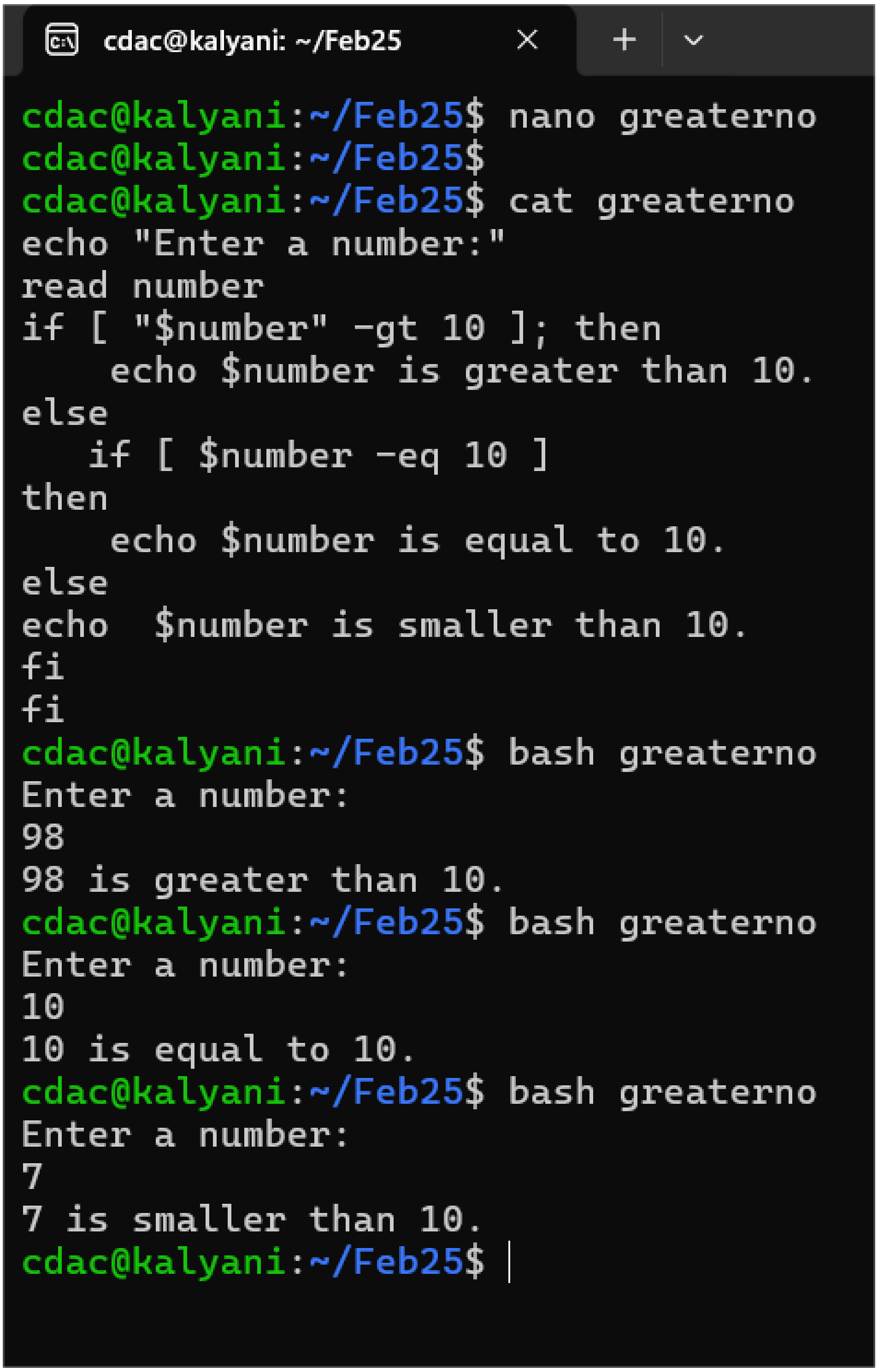
```
cdac@kalyani:~/Feb25$ nano while
cdac@kalyani:~/Feb25$ cat while
a=1
while [ $a -lt 6 ]
do
 echo $a
a=`expr $a + 1`
done
cdac@kalyani:~/Feb25$ bash while
1
2
3
4
5
cdac@kalyani:~/Feb25$
```

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".



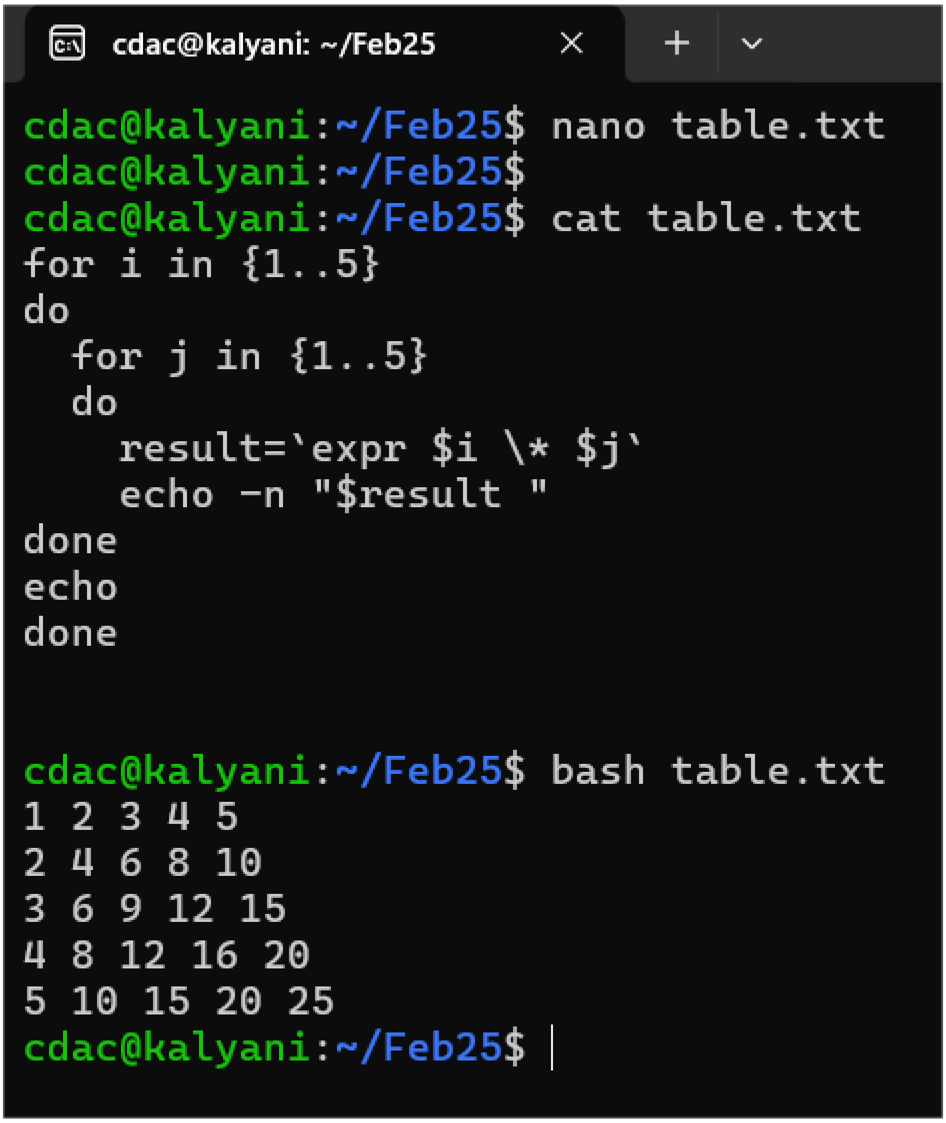
```
cdac@kalyani:~/Feb25$ nano filecheck.txt
cdac@kalyani:~/Feb25$ cat filecheck.txt
if [ -e file.txt ]
then
echo "File Exit"
else
echo "File does't exit"
fi
cdac@kalyani:~/Feb25$ bash filecheck.txt
File does't exit
cdac@kalyani:~/Feb25$
```

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@kalyani:~/Feb25$ nano greaterno
cdac@kalyani:~/Feb25$
cdac@kalyani:~/Feb25$ cat greaterno
echo "Enter a number:"
read number
if [ "$number" -gt 10 ]; then
    echo $number is greater than 10.
else
   if [ $number -eq 10 ]
then
    echo $number is equal to 10.
else
echo  $number is smaller than 10.
fi
fi
cdac@kalyani:~/Feb25$ bash greaterno
Enter a number:
98
98 is greater than 10.
cdac@kalyani:~/Feb25$ bash greaterno
Enter a number:
10
10 is equal to 10.
cdac@kalyani:~/Feb25$ bash greaterno
Enter a number:
7
7 is smaller than 10.
cdac@kalyani:~/Feb25$
```

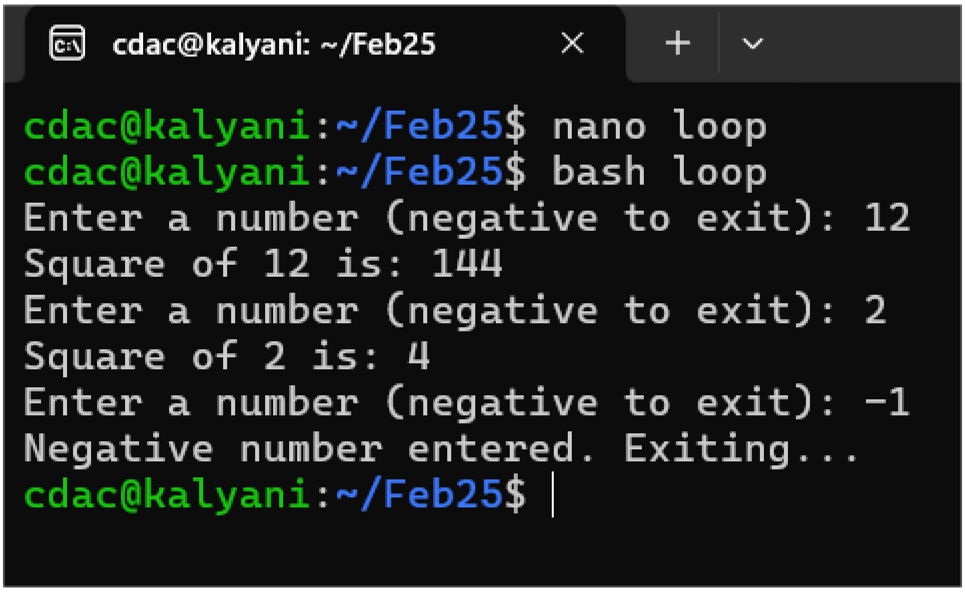
10. Write a shell script that uses nested for loops to print a multiplication table for numbers from 1 to 5. The output should be formatted nicely, with each row representing a number and each column representing the multiplication result for that number.



```
cdac@kalyani:~/Feb25$ nano table.txt
cdac@kalyani:~/Feb25$
cdac@kalyani:~/Feb25$ cat table.txt
for i in {1..5}
do
  for j in {1..5}
  do
    result=`expr $i \* $j`
    echo -n "$result "
done
echo
done


cdac@kalyani:~/Feb25$ bash table.txt
1 2 3 4 5
2 4 6 8 10
3 6 9 12 15
4 8 12 16 20
5 10 15 20 25
cdac@kalyani:~/Feb25$
```

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.



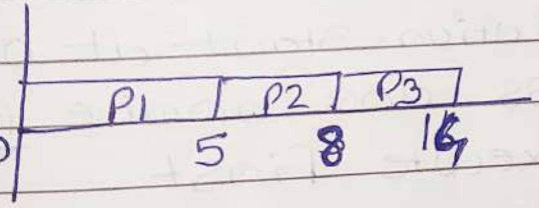
```
cdac@kalyani:~/Feb25$ nano loop
cdac@kalyani:~/Feb25$ bash loop
Enter a number (negative to exit): 12
Square of 12 is: 144
Enter a number (negative to exit): 2
Square of 2 is: 4
Enter a number (negative to exit): -1
Negative number entered. Exiting...
cdac@kalyani:~/Feb25$
```

1. consider the following Process with arrival times and burst times.

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

Process	AT	BT	W.T	RT (Response time)
P1	0	5	0	0
P2	1	3	4	5
P3	2	6	6	8

Gantt chart



* Waiting time = Resonse time - Arrival time

* Average waiting time = $\frac{\text{Total of W.t}}{\text{no. of Process}}$

$= \frac{0+4+6}{3}$

$= \frac{10}{3}$

$= 3.33$

∴ Average waiting time.

2. Calculate the Average Turn around time using Shortest Job First (SJF) Scheduling.

Process	Arrival time	Burst time	RT	WT	TAT
P1	0	3	0	0	3
P2	1	~~3~~ 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

Step1:- Execution order (SJF)

- The CPU always selects the Process with SJF from available Process.
- when execution Start at 0 How many Process are arrive P1

1) then P1 execute first

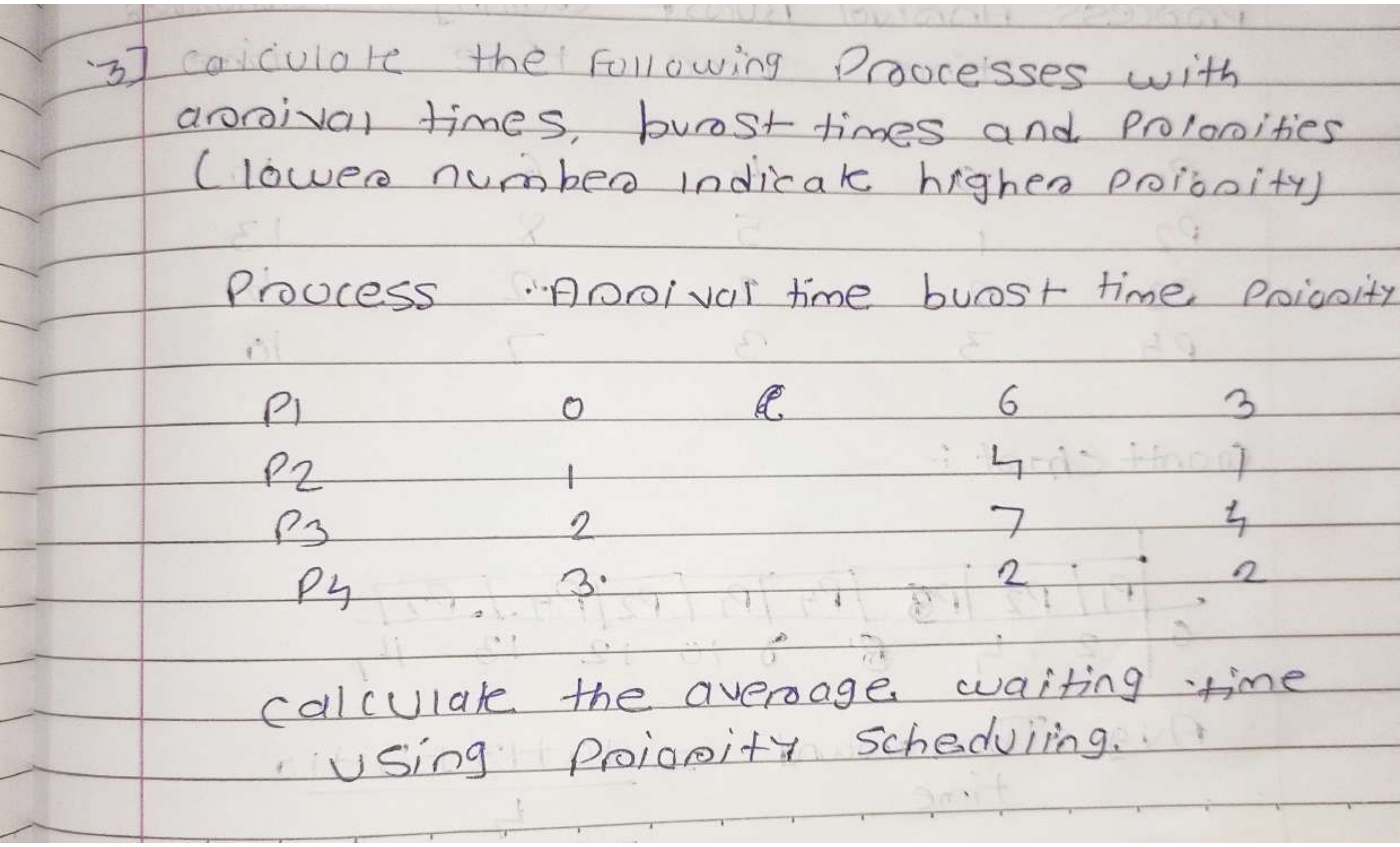
- Then execution time become 3 at 3 how many Process are available P2 P3, P4 we have three Process so we are using SJF so less burst time choose this Process

2) P3 have less burst time

∴ Avg turnaround time $= \frac{3+12+2+5}{4}$

$= \frac{22}{4}$

$= 5.5$

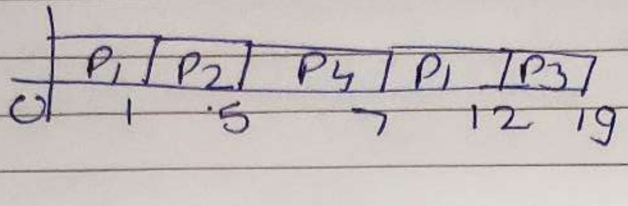


3] calculate the following Processes with arrival times, burst times and Priorities (lower number indicate higher priority)

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.

Gantt chart (preemptive)



waiting time = Response time - Arrival time

∴ Avg waiting time = $\frac{\text{total of waiting time}}{\text{no. of process}}$

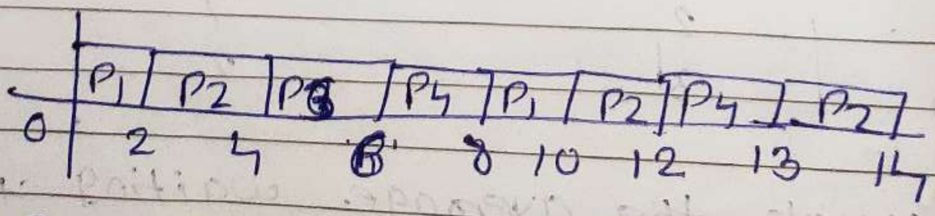
$= \frac{18}{4}$

$= 4.5$

4] Round Robin

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 :-



Avg Turnaround time $= \frac{10+13+4+10}{4}$