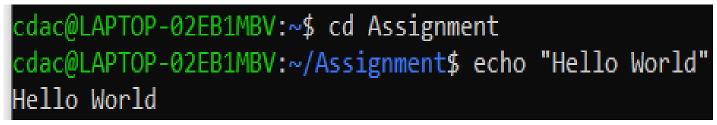
CDAC MUMBAI

Concepts of Operating System
Assignment 2

Part A

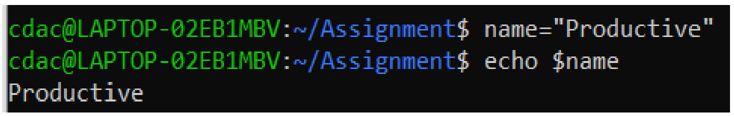
What will the following commands do?

- echo "Hello, World!"
  Ans : Prints the text "Hello, World!"



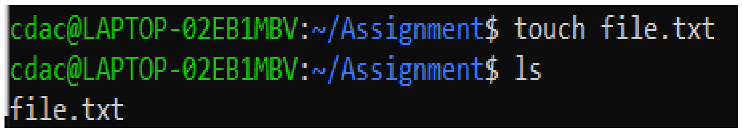
```
cdac@LAPTOP-02EB1MBV:~$ cd Assignment
cdac@LAPTOP-02EB1MBV:~/Assignment$ echo "Hello World"
Hello World
```

- name="Productive"
  Ans: Assigns to the string "Productive" to the variable name



```
cdac@LAPTOP-02EB1MBV:~/Assignment$ name="Productive"
cdac@LAPTOP-02EB1MBV:~/Assignment$ echo $name
Productive
```

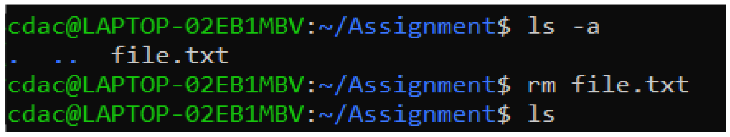
- touch file.txt
  Ans : touch is use to create a file.



```
cdac@LAPTOP-02EB1MBV:~/Assignment$ touch file.txt
cdac@LAPTOP-02EB1MBV:~/Assignment$ ls
file.txt
```

- ls -a

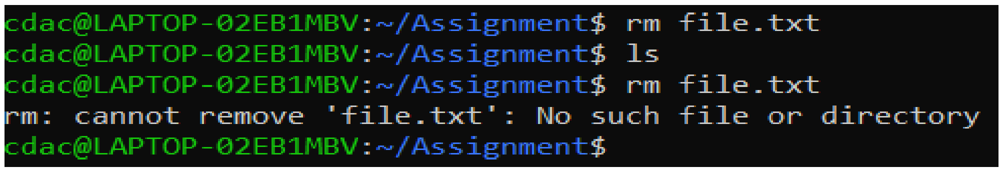
Ans: Lists all files and directories in the current directory, including hidden files (those starting with dot.)



```
cdac@LAPTOP-02EB1MBV:~/Assignment$ ls -a
.  ..  file.txt
cdac@LAPTOP-02EB1MBV:~/Assignment$ rm file.txt
cdac@LAPTOP-02EB1MBV:~/Assignment$ ls
```

- rm file.txt

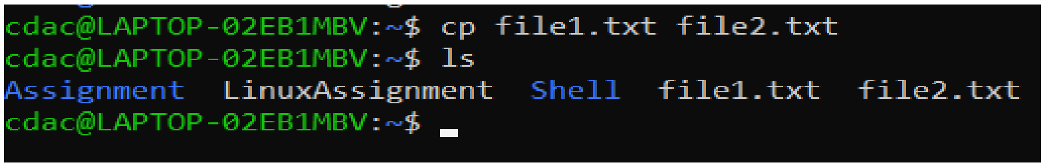
Ans: Deletes the file named file.txt



```
cdac@LAPTOP-02EB1MBV:~/Assignment$ rm file.txt
cdac@LAPTOP-02EB1MBV:~/Assignment$ ls
cdac@LAPTOP-02EB1MBV:~/Assignment$ rm file.txt
rm: cannot remove 'file.txt': No such file or directory
cdac@LAPTOP-02EB1MBV:~/Assignment$
```

- cp file1.txt file2.txt

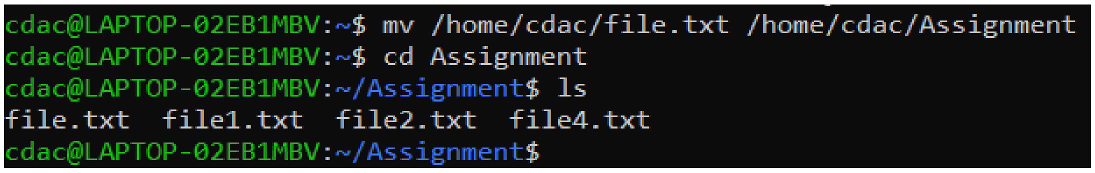
Ans: Command used to Copy file1.txt to file2.txt. If file2.txt exists, it will be overwritten the existing file.



```
cdac@LAPTOP-02EB1MBV:~$ cp file1.txt file2.txt
cdac@LAPTOP-02EB1MBV:~$ ls
Assignment  LinuxAssignment  Shell  file1.txt  file2.txt
cdac@LAPTOP-02EB1MBV:~$
```

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

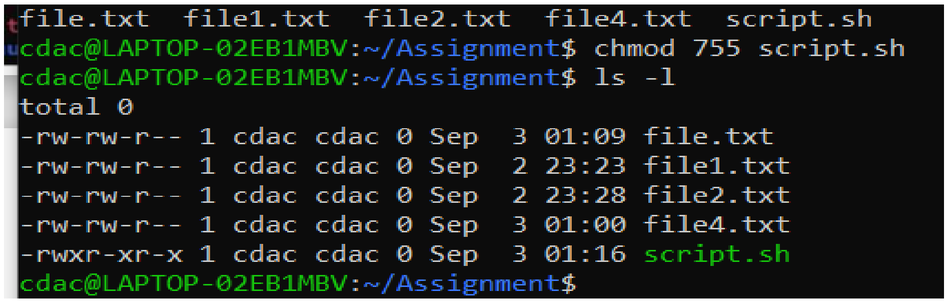
Ans: Moves file.txt to the specified directory



```
cdac@LAPTOP-02EB1MBV:~$ mv /home/cdac/file.txt /home/cdac/Assignment
cdac@LAPTOP-02EB1MBV:~$ cd Assignment
cdac@LAPTOP-02EB1MBV:~/Assignment$ ls
file.txt  file1.txt  file2.txt  file4.txt
cdac@LAPTOP-02EB1MBV:~/Assignment$
```

- chmod 755 script.sh

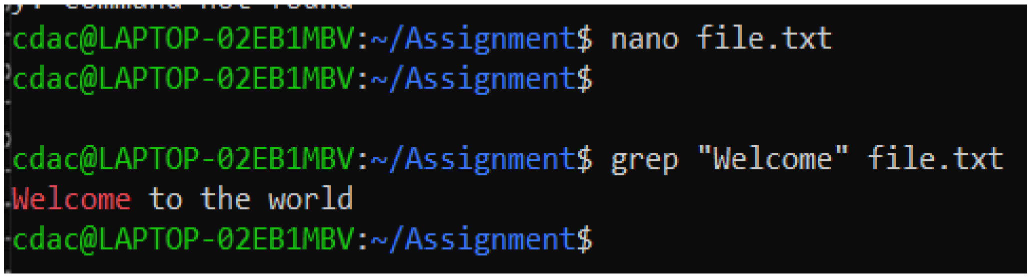
Ans: The given command Changes the permissions of file.txt to 755, giving the owner full read, write, and execute permissions, and giving others read and execute permissions.



```
file.txt  file1.txt  file2.txt  file4.txt  script.sh
cdac@LAPTOP-02EB1MBV:~/Assignment$ chmod 755 script.sh
cdac@LAPTOP-02EB1MBV:~/Assignment$ ls -l
total 0
-rw-rw-r-- 1 cdac cdac 0 Sep  3 01:09 file.txt
-rw-rw-r-- 1 cdac cdac 0 Sep  2 23:23 file1.txt
-rw-rw-r-- 1 cdac cdac 0 Sep  2 23:28 file2.txt
-rw-rw-r-- 1 cdac cdac 0 Sep  3 01:00 file4.txt
-rwxr-xr-x 1 cdac cdac 0 Sep  3 01:16 script.sh
cdac@LAPTOP-02EB1MBV:~/Assignment$
```

- grep "pattern" file.txt

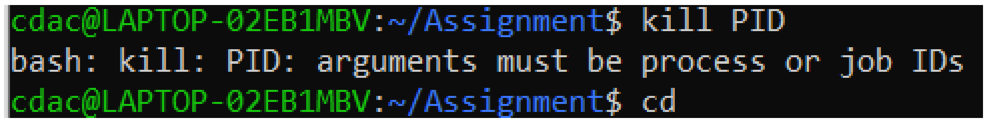
Ans: Grep command searches for the string of "characters" in file.txt and displays all matching lines.



```
cdac@LAPTOP-02EB1MBV:~/Assignment$ nano file.txt
cdac@LAPTOP-02EB1MBV:~/Assignment$

cdac@LAPTOP-02EB1MBV:~/Assignment$ grep "Welcome" file.txt
Welcome to the world
cdac@LAPTOP-02EB1MBV:~/Assignment$
```

- kill PID

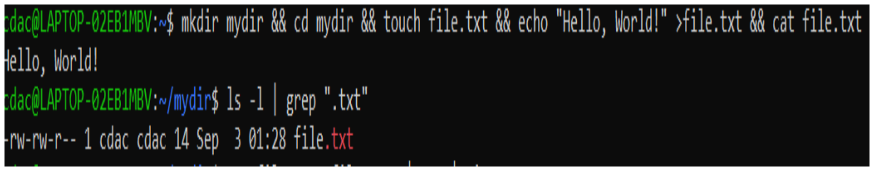
Ans : Terminates the process with the specified Process ID (PID)



```
cdac@LAPTOP-02EB1MBV:~/Assignment$ kill PID
bash: kill: PID: arguments must be process or job IDs
cdac@LAPTOP-02EB1MBV:~/Assignment$ cd
```

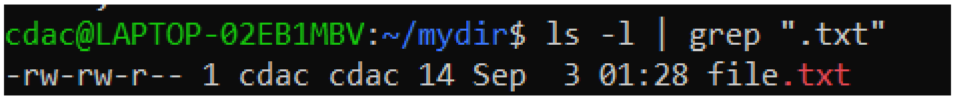
- mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt

Ans: The series of instructions concatenated with &&, mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt, guarantees that each command is executed only in the event that the preceding one is run.



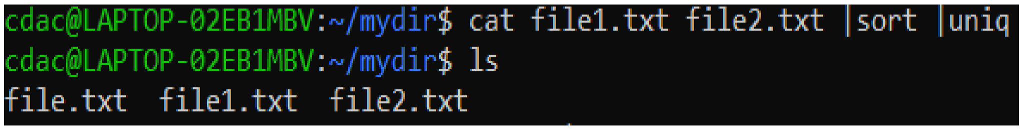
```
cdac@LAPTOP-02EB1MBV:~$ mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" >file.txt && cat file.txt
Hello, World!
cdac@LAPTOP-02EB1MBV:~/mydir$ ls -l | grep ".txt"
-rw-rw-r-- 1 cdac cdac 14 Sep  3 01:28 file.txt
```

- ls -l | grep ".txt"
  Ans: Lists files in the current directory in long format (-l) and filters the output to show only those with .txt in their names.



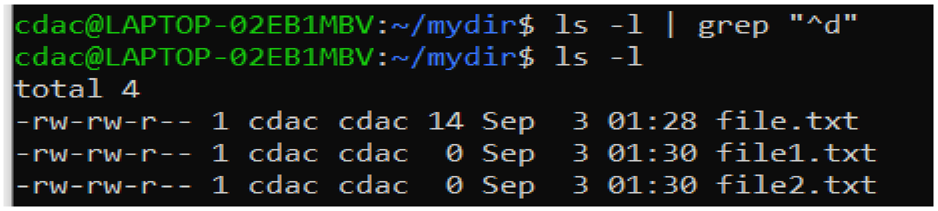
```
cdac@LAPTOP-02EB1MBV:~/mydir$ ls -l | grep ".txt"
-rw-rw-r-- 1 cdac cdac 14 Sep  3 01:28 file.txt
```

- cat file1.txt file2.txt | sort | uniq
  Ans: Concatenates file1.txt and file2.txt, sorts the combined output, and removes duplicate lines



```
cdac@LAPTOP-02EB1MBV:~/mydir$ cat file1.txt file2.txt |sort |uniq
cdac@LAPTOP-02EB1MBV:~/mydir$ ls
file.txt  file1.txt  file2.txt
```

- ls -l | grep "^d"
  Ans: Uses the lengthy format to list files in the current directory, then filters the output to display only directories (denoted by a "d" in the permissions column).

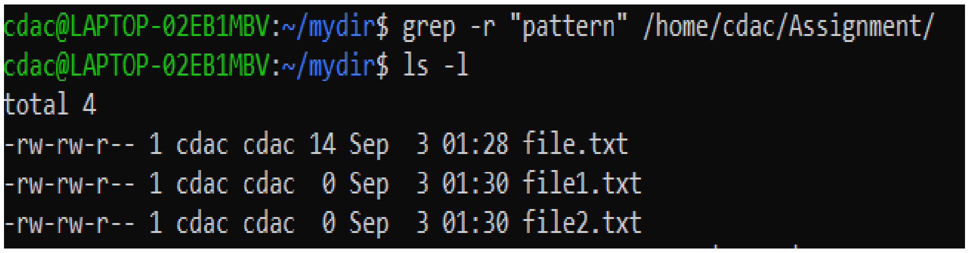


```
cdac@LAPTOP-02EB1MBV:~/mydir$ ls -l | grep "^d"
cdac@LAPTOP-02EB1MBV:~/mydir$ ls -l
total 4
-rw-rw-r-- 1 cdac cdac 14 Sep  3 01:28 file.txt
-rw-rw-r-- 1 cdac cdac  0 Sep  3 01:30 file1.txt
-rw-rw-r-- 1 cdac cdac  0 Sep  3 01:30 file2.txt
```

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

  Ans: The command grep -r "pattern" /path/to/directory/ is used to search for a specific text pattern within all files in a directory and its subdirectories.

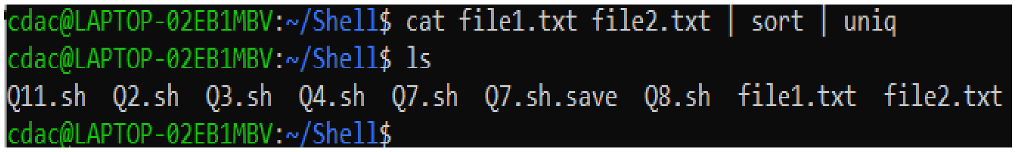
  Command: grep -r "pattern" /home/cdac/Assignment/



```
cdac@LAPTOP-02EB1MBV:~/mydir$ grep -r "pattern" /home/cdac/Assignment/
cdac@LAPTOP-02EB1MBV:~/mydir$ ls -l
total 4
-rw-rw-r-- 1 cdac cdac 14 Sep  3 01:28 file.txt
-rw-rw-r-- 1 cdac cdac  0 Sep  3 01:30 file1.txt
-rw-rw-r-- 1 cdac cdac  0 Sep  3 01:30 file2.txt
```

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

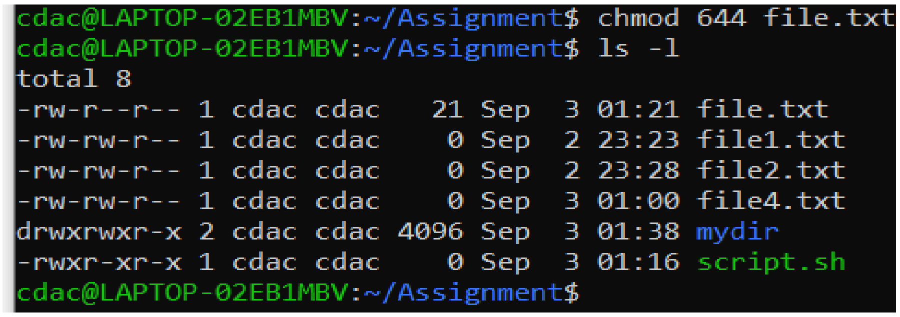
  Ans: Concatenates file1.txt and file2.txt, sorts the combined output, and displays only duplicate lines with the help of sort and uniq commands.



```
cdac@LAPTOP-02EB1MBV:~/Shell$ cat file1.txt file2.txt | sort | uniq
cdac@LAPTOP-02EB1MBV:~/Shell$ ls
Q11.sh  Q2.sh  Q3.sh  Q4.sh  Q7.sh  Q7.sh.save  Q8.sh  file1.txt  file2.txt
cdac@LAPTOP-02EB1MBV:~/Shell$
```

- chmod 644 file.txt

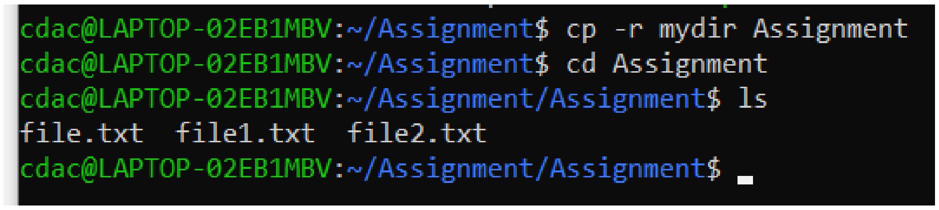
  Ans: the Chmod 644 Changes the permissions of file.txt to 644, giving the owner read and write permissions, and giving others read-only permissions



```
cdac@LAPTOP-02EB1MBV:~/Assignment$ chmod 644 file.txt
cdac@LAPTOP-02EB1MBV:~/Assignment$ ls -l
total 8
-rw-r--r-- 1 cdac cdac   21 Sep  3 01:21 file.txt
-rw-rw-r-- 1 cdac cdac    0 Sep  2 23:23 file1.txt
-rw-rw-r-- 1 cdac cdac    0 Sep  2 23:28 file2.txt
-rw-rw-r-- 1 cdac cdac    0 Sep  3 01:00 file4.txt
drwxrwxr-x 2 cdac cdac 4096 Sep  3 01:38 mydir
-rwxr-xr-x 1 cdac cdac    0 Sep  3 01:16 script.sh
cdac@LAPTOP-02EB1MBV:~/Assignment$
```

- cp -r source_directory destination_directory

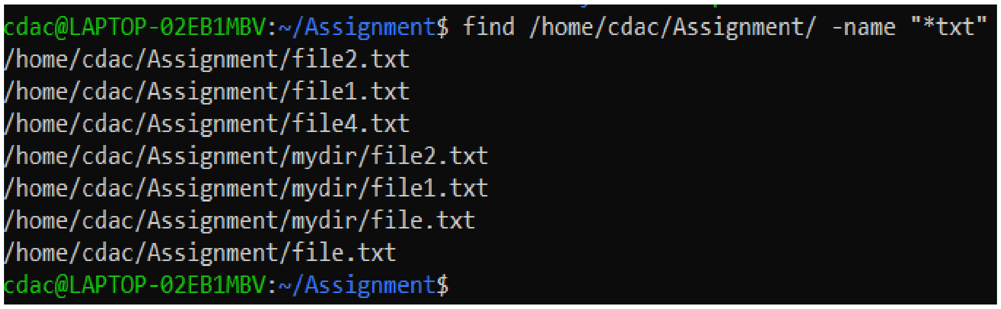
  Ans: The command "cp -r source_directory destination_directory" copies the whole source directory along with all of its contents, then stores the copied copy in destination_directory.



```
cdac@LAPTOP-02EB1MBV:~/Assignment$ cp -r mydir Assignment
cdac@LAPTOP-02EB1MBV:~/Assignment$ cd Assignment
cdac@LAPTOP-02EB1MBV:~/Assignment/Assignment$ ls
file.txt  file1.txt  file2.txt
cdac@LAPTOP-02EB1MBV:~/Assignment/Assignment$
```

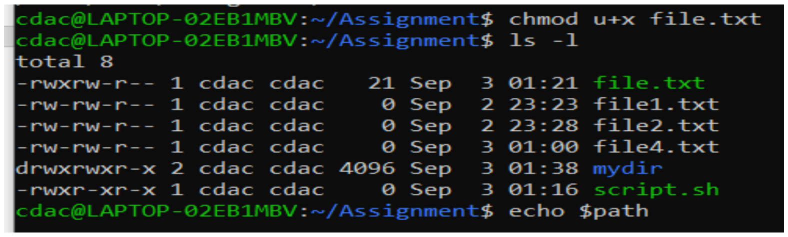
- find /path/to/search -name "*.txt"

  Ans: Searches for all files with .txt extension within the specified directory and its subdirectories.



```
cdac@LAPTOP-02EB1MBV:~/Assignment$ find /home/cdac/Assignment/ -name "*txt"
/home/cdac/Assignment/file2.txt
/home/cdac/Assignment/file1.txt
/home/cdac/Assignment/file4.txt
/home/cdac/Assignment/mydir/file2.txt
/home/cdac/Assignment/mydir/file1.txt
/home/cdac/Assignment/mydir/file.txt
/home/cdac/Assignment/file.txt
cdac@LAPTOP-02EB1MBV:~/Assignment$
```

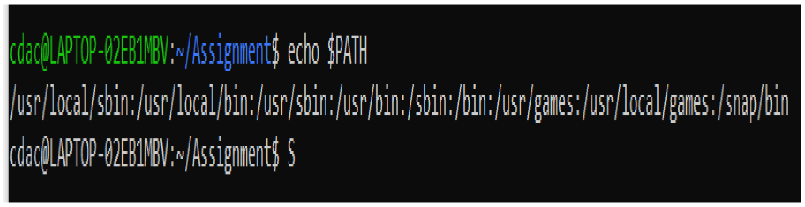
- chmod u+x file.txt
  Ans: Adds execute permission for the (user) of file.txt



```
cdac@LAPTOP-02EB1MBV:~/Assignment$ chmod u+x file.txt
cdac@LAPTOP-02EB1MBV:~/Assignment$ ls -l
total 8
-rwxrw-r-- 1 cdac cdac   21 Sep  3 01:21 file.txt
-rw-rw-r-- 1 cdac cdac    0 Sep  2 23:23 file1.txt
-rw-rw-r-- 1 cdac cdac    0 Sep  2 23:28 file2.txt
-rw-rw-r-- 1 cdac cdac    0 Sep  3 01:00 file4.txt
drwxrwxr-x 2 cdac cdac 4096 Sep  3 01:38 mydir
-rwxr-xr-x 1 cdac cdac    0 Sep  3 01:16 script.sh
cdac@LAPTOP-02EB1MBV:~/Assignment$ echo $path
```

- echo $PATH

  Ans: When we used this command, echo $PATH displayed the list of folders (or path) that your computer searches for software to launch.



```
cdac@LAPTOP-02EB1MBV:~/Assignment$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
cdac@LAPTOP-02EB1MBV:~/Assignment$ $
```

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.
Ans: chmod

2. **cpy** is used to copy files and directories.
Ans: cp

3. **mkfile** is used to create a new file.
Ans: touch

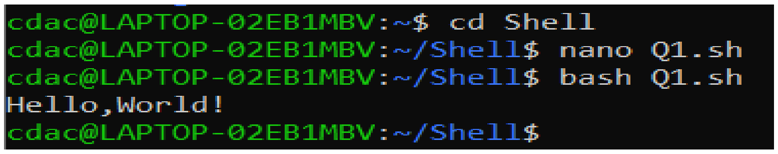
4. **catx** is used to concatenate files.
Ans: cat

5. **rn** is used to rename files.
Ans: mv

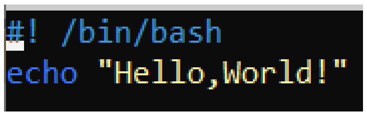
Part C

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

Command: nano Q1.sh
bash Q1.sh



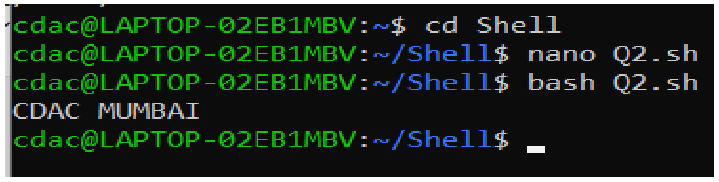
```
cdac@LAPTOP-02EB1MBV:~$ cd Shell
cdac@LAPTOP-02EB1MBV:~/Shell$ nano Q1.sh
cdac@LAPTOP-02EB1MBV:~/Shell$ bash Q1.sh
Hello,World!
cdac@LAPTOP-02EB1MBV:~/Shell$
```



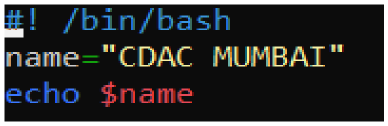
```
#! /bin/bash
echo "Hello,World!"
```

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

Command: nano Q2.sh
bash Q2.sh



```
cdac@LAPTOP-02EB1MBV:~$ cd Shell
cdac@LAPTOP-02EB1MBV:~/Shell$ nano Q2.sh
cdac@LAPTOP-02EB1MBV:~/Shell$ bash Q2.sh
CDAC MUMBAI
cdac@LAPTOP-02EB1MBV:~/Shell$
```

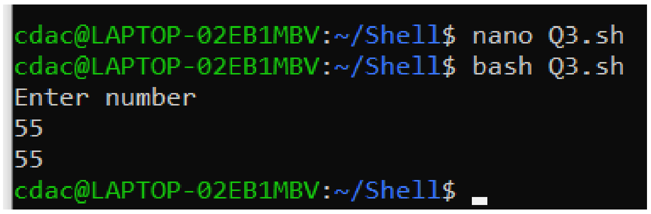


```
#! /bin/bash
name="CDAC MUMBAI"
echo $name
```

Question 3: Write a shell script that takes a number as input from the user and prints it.
Command: nano Q3.sh
bash Q3.sh

This command used to create file and for shell scripting

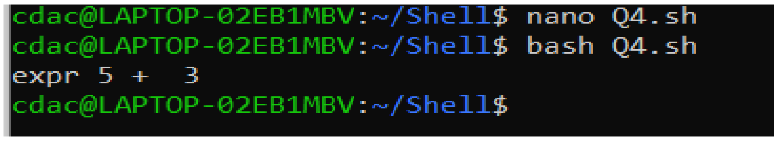
```
#!/bin/bash
echo "Enter number"
read number
echo $number
```



```
cdac@LAPTOP-02EB1MBV:~/Shell$ nano Q3.sh
cdac@LAPTOP-02EB1MBV:~/Shell$ bash Q3.sh
Enter number
55
55
cdac@LAPTOP-02EB1MBV:~/Shell$
```

Question 4: Write a shell script that performs addition of two numbers (e.g., 5 and 3) and prints the result.
Command: nano Q4.sh
bash Q4.sh
Script that perform addition of two numbers

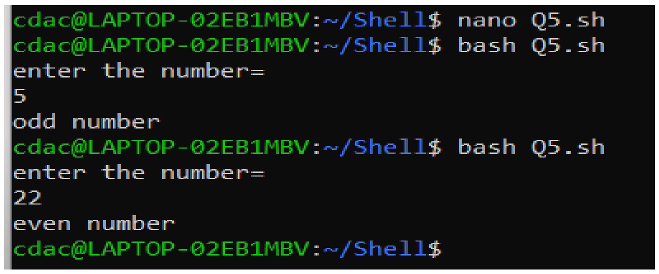
```
#!/bin/nash
a=4
b=7
c=$[a+b]
echo $c
```



```
cdac@LAPTOP-02EB1MBV:~/Shell$ nano Q4.sh
cdac@LAPTOP-02EB1MBV:~/Shell$ bash Q4.sh
expr 5 +  3
cdac@LAPTOP-02EB1MBV:~/Shell$
```

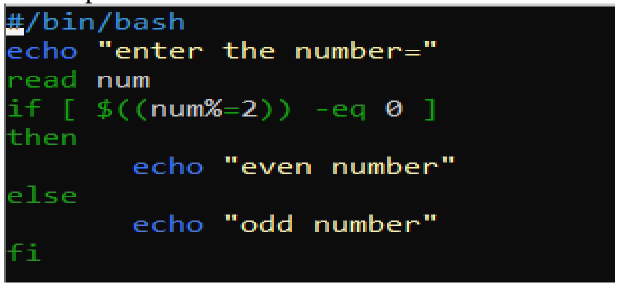
Question 5: Write a shell script that takes a number as input and prints "Even" if it is even, otherwise prints "Odd".

Command: nano Q5.sh
bash Q5.sh



```
cdac@LAPTOP-02EB1MBV:~/Shell$ nano Q5.sh
cdac@LAPTOP-02EB1MBV:~/Shell$ bash Q5.sh
enter the number=
5
odd number
cdac@LAPTOP-02EB1MBV:~/Shell$ bash Q5.sh
enter the number=
22
even number
cdac@LAPTOP-02EB1MBV:~/Shell$
```

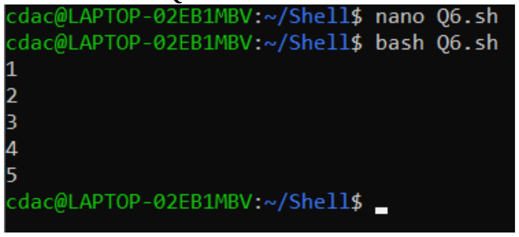
Shell script foe even and odd



```
#/bin/bash
echo "enter the number="
read num
if [ $((num%=2)) -eq 0 ]
then
        echo "even number"
else
        echo "odd number"
fi
```

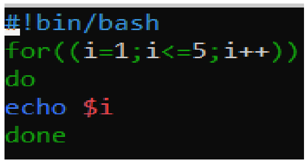
Question 6: Write a shell script that uses a for loop to print numbers from 1 to 5.

Command: nano Q6.sh
          bash Q6.sh



```
cdac@LAPTOP-02EB1MBV:~/Shell$ nano Q6.sh
cdac@LAPTOP-02EB1MBV:~/Shell$ bash Q6.sh
1
2
3
4
5
cdac@LAPTOP-02EB1MBV:~/Shell$ _
```

Shell Script that uses a for loop to print numbers from 1 to 5

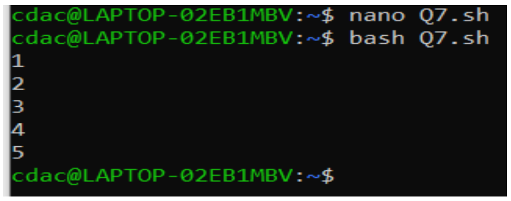


```
#!bin/bash
for((i=1;i<=5;i++))
do
echo $i
done
```

Question 7: Write a shell script that uses a while loop to print numbers from 1 to 5.
Command: nano Q7.sh
         bash Q7.sh

In shell script while loop is used to print from number from 1 to 5

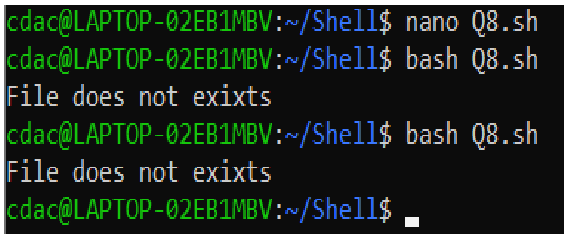
```
 #!/bin/bash
 i=1

while [ $i -le 5 ]
do
echo $i
i=$(($i+1))

done
```



```
cdac@LAPTOP-02EB1MBV:~$ nano Q7.sh
cdac@LAPTOP-02EB1MBV:~$ bash Q7.sh
1
2
3
4
5
cdac@LAPTOP-02EB1MBV:~$
```

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".
Command: nano Q8.sh
bash Q8.sh
Script:

```
 #!/bin/bash
 f="h.sh"
if [ -f "$f" ]
then
echo "File exixts"
else
echo "File does not exixts"
fi
```

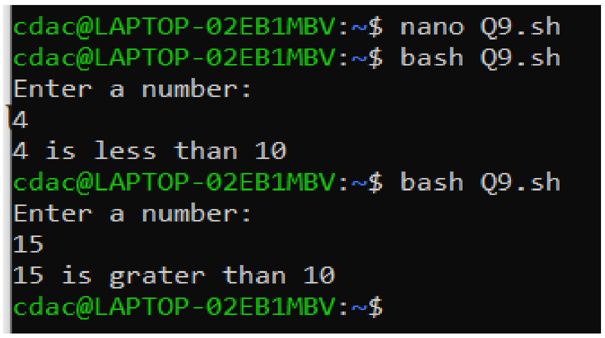


```
cdac@LAPTOP-02EB1MBV:~/Shell$ nano Q8.sh
cdac@LAPTOP-02EB1MBV:~/Shell$ bash Q8.sh
File does not exixts
cdac@LAPTOP-02EB1MBV:~/Shell$ bash Q8.sh
File does not exixts
cdac@LAPTOP-02EB1MBV:~/Shell$
```

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.
Command: nano Q9.sh
bash Q9.sh

Shell script:

```
      #!/bin/bash
 echo "Enter a number:"
 read number
if [ $number -gt 10 ]
then
echo "$number is grater than 10"
else
echo "$number is less than 10"
 fi
```



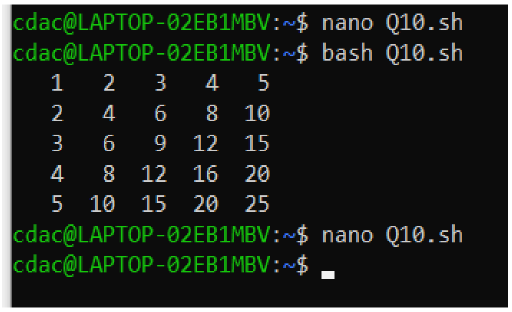
```
cdac@LAPTOP-02EB1MBV:~$ nano Q9.sh
cdac@LAPTOP-02EB1MBV:~$ bash Q9.sh
Enter a number:
4
4 is less than 10
cdac@LAPTOP-02EB1MBV:~$ bash Q9.sh
Enter a number:
15
15 is grater than 10
cdac@LAPTOP-02EB1MBV:~$
```

Question 10: Write a shell script that uses nested for loops to print a multiplication table for numbersfrom 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.

Command: nano Q10.sh
bash Q10.sh

Shell Script:

```
#!/bin/bash
for ((i=1; i<=5; i++))
do
for ((j=1; j<=5; j++))
do
printf "%4d" "$((i*j))"
done
echo
done
```



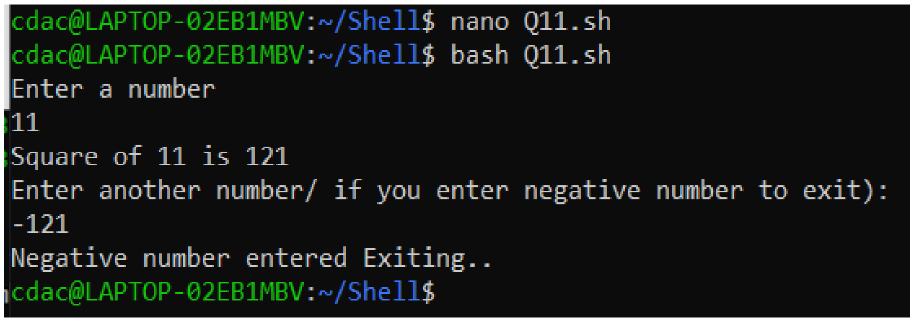
```
cdac@LAPTOP-02EB1MBV:~$ nano Q10.sh
cdac@LAPTOP-02EB1MBV:~$ bash Q10.sh
   1   2   3   4   5
   2   4   6   8  10
   3   6   9  12  15
   4   8  12  16  20
   5  10  15  20  25
cdac@LAPTOP-02EB1MBV:~$ nano Q10.sh
cdac@LAPTOP-02EB1MBV:~$
```

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

```
Command: nano Q11.sh
         bash Q11.sh
```

Shell script:

```
 #!/bin/bash
 echo "Enter a number"
 read num
 while [ "$num" -gt 0 ]
 do
 sq=$((num * num))
echo "Square of $num is $sq"

echo "Enter another number/ if you enter negative number to exit):"
 read num
 done
 echo "Negative number entered Exiting.."
```



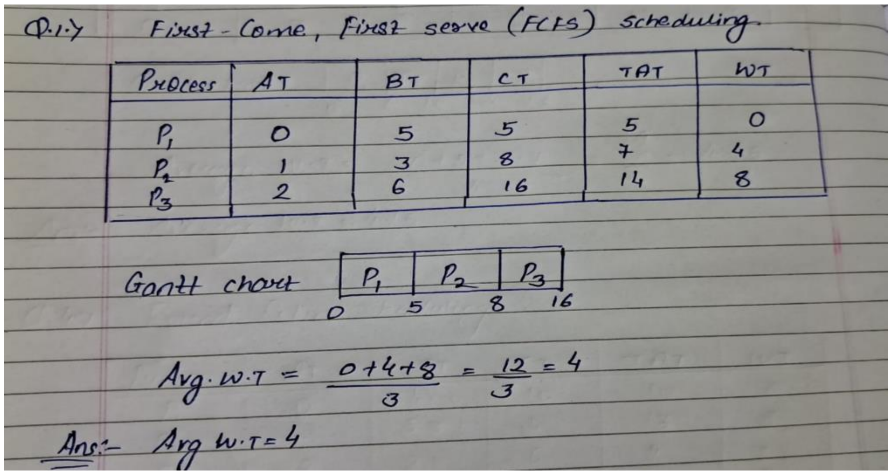
```
cdac@LAPTOP-02EB1MBV:~/Shell$ nano Q11.sh
cdac@LAPTOP-02EB1MBV:~/Shell$ bash Q11.sh
Enter a number
11
Square of 11 is 121
Enter another number/ if you enter negative number to exit):
-121
Negative number entered Exiting..
cdac@LAPTOP-02EB1MBV:~/Shell$
```

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.



Q.1.) First - Come, First serve (FCFS) scheduling.

Process	AT	BT	CT	TAT	WT
P_1	0	5	5	5	0
P_2	1	3	8	7	4
P_3	2	6	16	14	8

Gantt chart

P_1	P_2	P_3

0 5 8 16

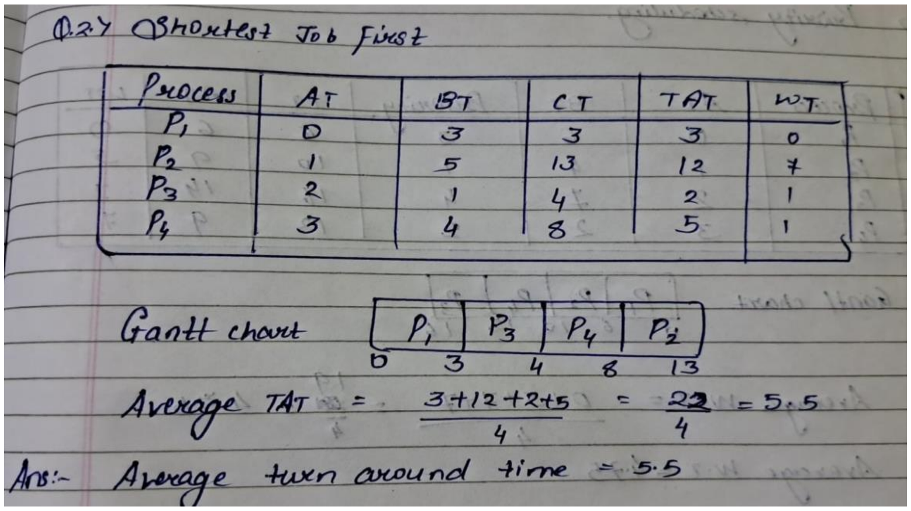
$$\text{Avg. W.T} = \frac{0+4+8}{3} = \frac{12}{3} = 4$$

Ans:- Avg W.T = 4

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.



Q.2.) Shortest Job First

Process	AT	BT	CT	TAT	WT
P_1	0	3	3	3	0
P_2	1	5	13	12	7
P_3	2	1	4	2	1
P_4	3	4	8	5	1

Gantt chart

P_1	P_3	P_4	P_2

0 3 4 8 13

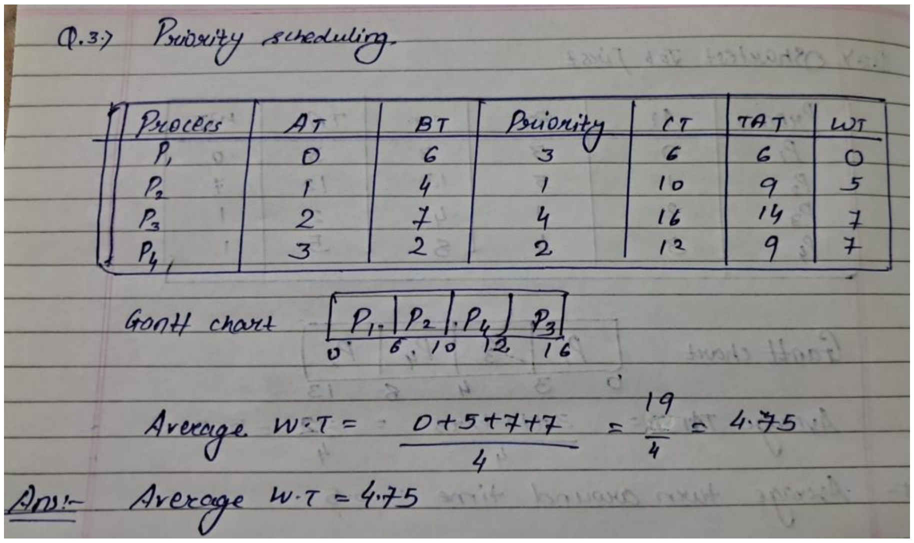
$$\text{Average TAT} = \frac{3+12+2+5}{4} = \frac{22}{4} = 5.5$$

Ans:- Average turn around time = 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



Q.3.) Priority scheduling.

Process	AT	BT	Priority	CT	TAT	WT
P_1	0	6	3	6	6	0
P_2	1	4	1	10	9	5
P_3	2	7	4	16	14	7
P_4	3	2	2	12	9	7

Gantt chart

P_1	P_2	P_4	P_3

0 6 10 12 16

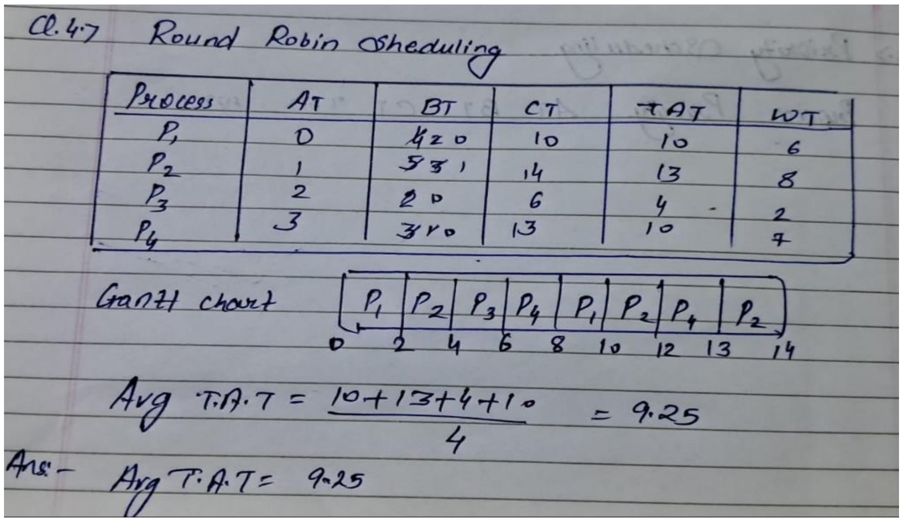
Average W.T = $\frac{0+5+7+7}{4} = \frac{19}{4} = 4.75$

Ans:- Average W.T = 4.75

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.



Q.4.) Round Robin Sheduling

Process	AT	BT	CT	TAT	WT
P_1	0	~~4~~ ~~2~~ 0	10	10	6
P_2	1	~~5~~ ~~3~~ 1	14	13	8
P_3	2	~~2~~ 0	6	4	2
P_4	3	~~3~~ ~~1~~ 0	13	10	7

Gantt chart

P_1	P_2	P_3	P_4	P_1	P_2	P_4	P_2

0 2 4 6 8 10 12 13 14

Avg T.A.T = $\frac{10+13+4+10}{4} = 9.25$

Ans:- Avg T.A.T = 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: After fork ( ) call

```
#include <stdio.h>

void main()
{

  int x = 5;
 fork();
x = x+1;
 printf("x = %d\n",x);
    }
```

Submission Guidelines:

- Document each step of your solution and any challenges faced.
- Upload it on your GitHub repository

Additional Tips:

- Experiment with different options and parameters of each command to explore their functionalities.
- This assignment is tailored to align with interview expectations, CCEE standards, and industry demands.
- If you complete this then your preparation will be skyrocketed.