CDAC MUMBAI

Concepts of Operating System

Assignment 2

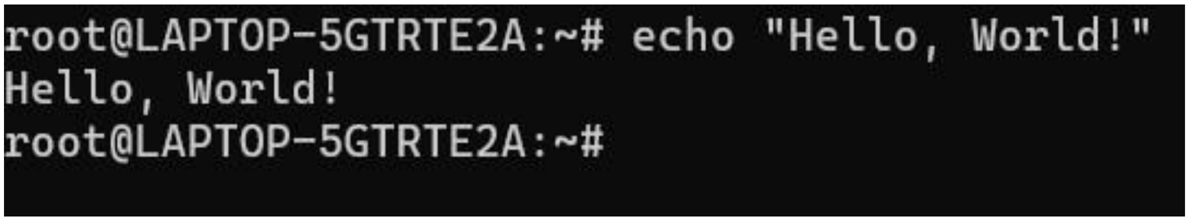
Part A

What will the following commands do?

• echo "Hello, World!"

Echo command is used to print any line, word, string etc

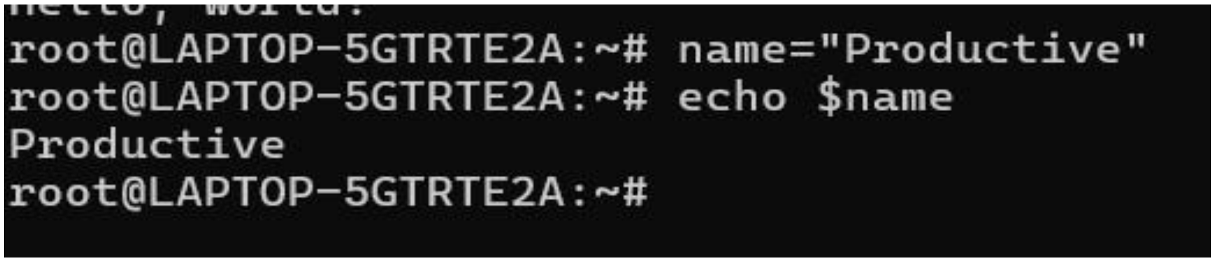
In the given command it will print Hello, World!



```
root@LAPTOP-5GTRTE2A:~# echo "Hello, World!"
Hello, World!
root@LAPTOP-5GTRTE2A:~#
```

• name="Productive"

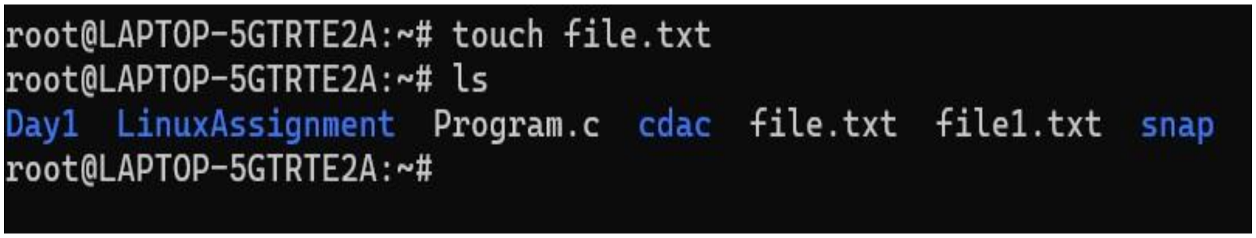
Here name is a variable which is used to assign a value.



```
root@LAPTOP-5GTRTE2A:~# name="Productive"
root@LAPTOP-5GTRTE2A:~# echo $name
Productive
root@LAPTOP-5GTRTE2A:~#
```

• touch file.txt

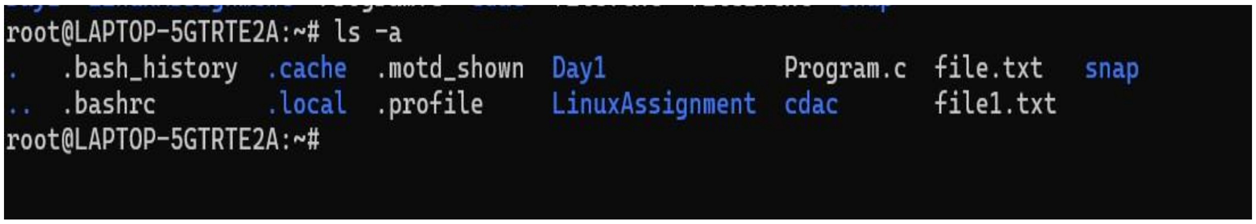
touch is used to create a new file.



```
root@LAPTOP-5GTRTE2A:~# touch file.txt
root@LAPTOP-5GTRTE2A:~# ls
Day1  LinuxAssignment  Program.c  cdac  file.txt  file1.txt  snap
root@LAPTOP-5GTRTE2A:~#
```

• ls -a

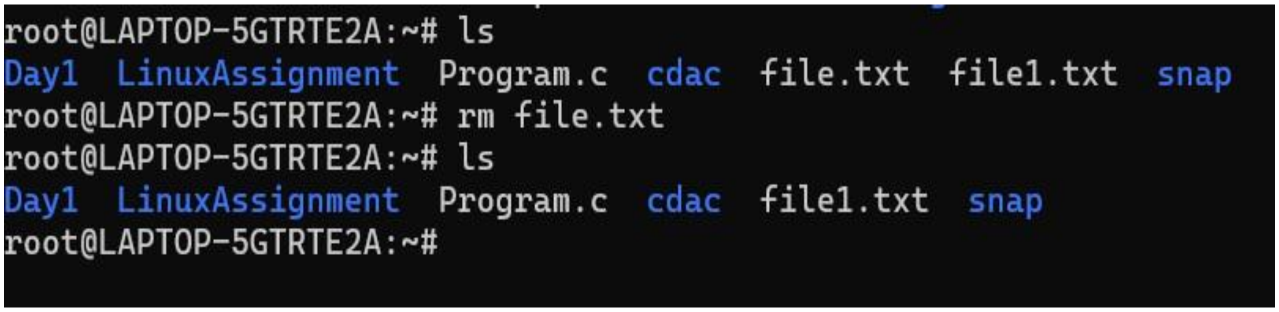
ls -a command is used to show the list of all files & directory including hidden files.



```
root@LAPTOP-5GTRTE2A:~# ls -a
.   .bash_history  .cache  .motd_shown  Day1             Program.c  file.txt   snap
..  .bashrc        .local  .profile     LinuxAssignment  cdac       file1.txt
root@LAPTOP-5GTRTE2A:~#
```

• rm file.txt

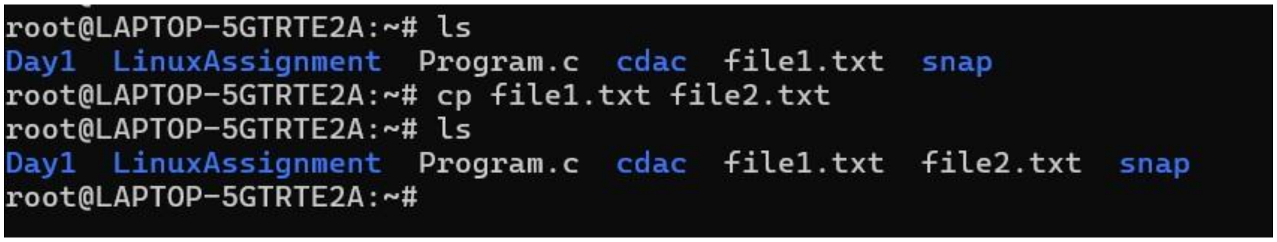
This command is used to remove a file.



```
root@LAPTOP-5GTRTE2A:~# ls
Day1  LinuxAssignment  Program.c  cdac  file.txt  file1.txt  snap
root@LAPTOP-5GTRTE2A:~# rm file.txt
root@LAPTOP-5GTRTE2A:~# ls
Day1  LinuxAssignment  Program.c  cdac  file1.txt  snap
root@LAPTOP-5GTRTE2A:~#
```

• cp file1.txt file2.txt

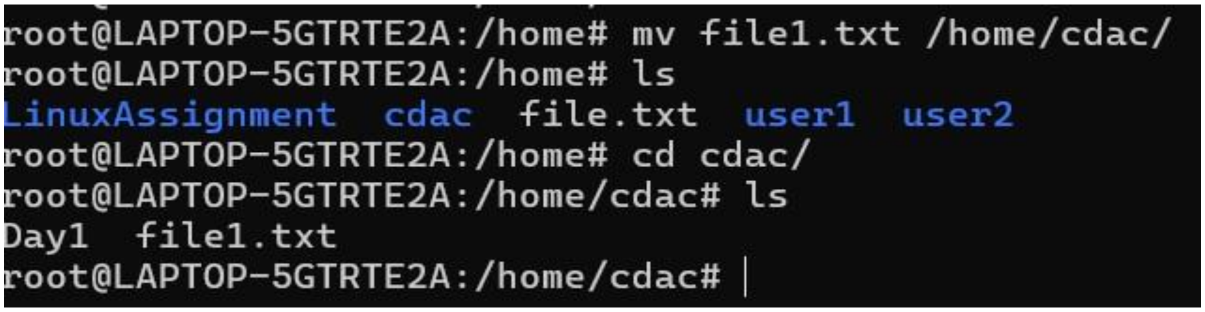
This command is used to copy file.



```
root@LAPTOP-5GTRTE2A:~# ls
Day1  LinuxAssignment  Program.c  cdac  file1.txt  snap
root@LAPTOP-5GTRTE2A:~# cp file1.txt file2.txt
root@LAPTOP-5GTRTE2A:~# ls
Day1  LinuxAssignment  Program.c  cdac  file1.txt  file2.txt  snap
root@LAPTOP-5GTRTE2A:~#
```

• mv file.txt /path/to/directory/

This command is used to move the file to the specific directory.



```
root@LAPTOP-5GTRTE2A:/home# mv file1.txt /home/cdac/
root@LAPTOP-5GTRTE2A:/home# ls
LinuxAssignment  cdac  file.txt  user1  user2
root@LAPTOP-5GTRTE2A:/home# cd cdac/
root@LAPTOP-5GTRTE2A:/home/cdac# ls
Day1  file1.txt
root@LAPTOP-5GTRTE2A:/home/cdac#
```

• chmod 755 script.sh

This command is used to change the permission of a file. Here 755 is a numbers to specify the permission.

Read = 4

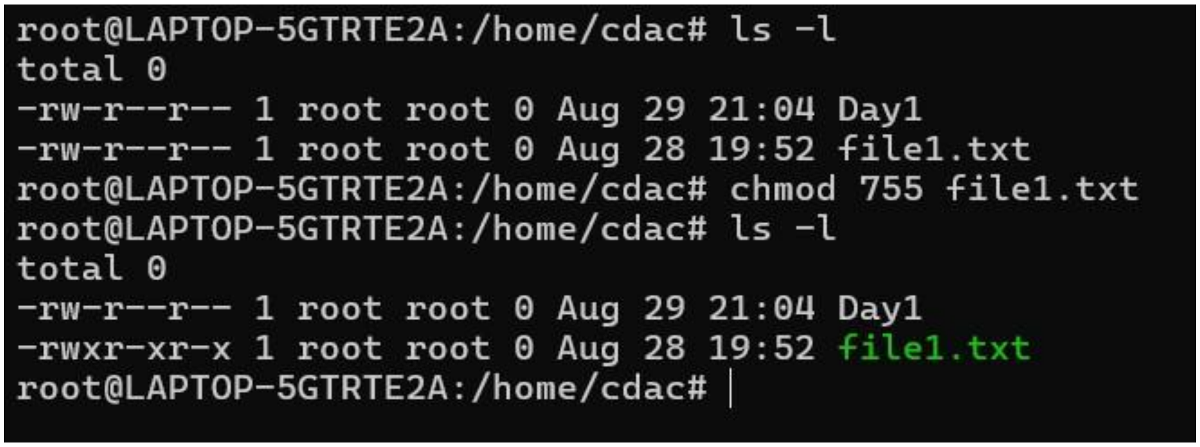
Write = 2

Execute = 1

7 => This number is for owner. 4+2+1 = 7 it means ower have all three permission to read , write & execute.

5 => This number is for group. 4+1 = 5 which means group have permission to read & execute.

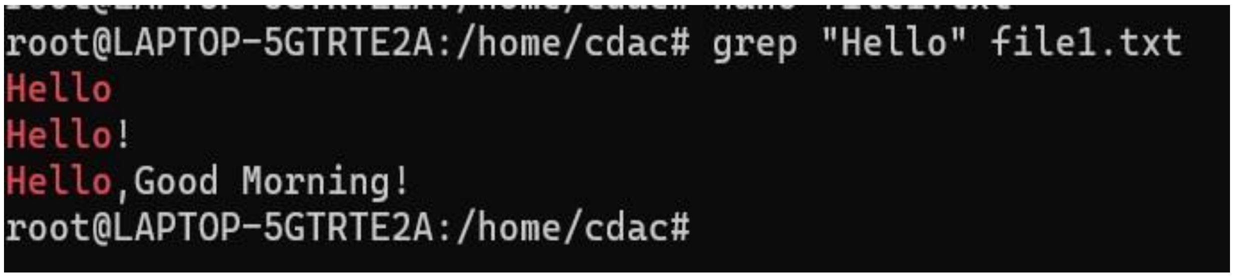
5 => This number is for all other. 4+1 = 5 which means others have permission to read & execute.



```
root@LAPTOP-5GTRTE2A:/home/cdac# ls -l
total 0
-rw-r--r-- 1 root root 0 Aug 29 21:04 Day1
-rw-r--r-- 1 root root 0 Aug 28 19:52 file1.txt
root@LAPTOP-5GTRTE2A:/home/cdac# chmod 755 file1.txt
root@LAPTOP-5GTRTE2A:/home/cdac# ls -l
total 0
-rw-r--r-- 1 root root 0 Aug 29 21:04 Day1
-rwxr-xr-x 1 root root 0 Aug 28 19:52 file1.txt
root@LAPTOP-5GTRTE2A:/home/cdac#
```

• grep "pattern" file.txt

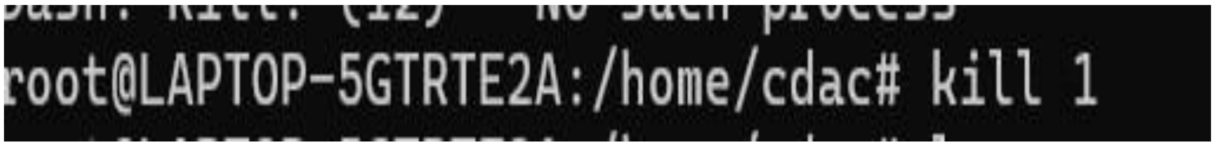
This command is used to search a specific pattern in a file.



```
root@LAPTOP-5GTRTE2A:/home/cdac# grep "Hello" file1.txt
Hello
Hello!
Hello,Good Morning!
root@LAPTOP-5GTRTE2A:/home/cdac#
```

• kill PID

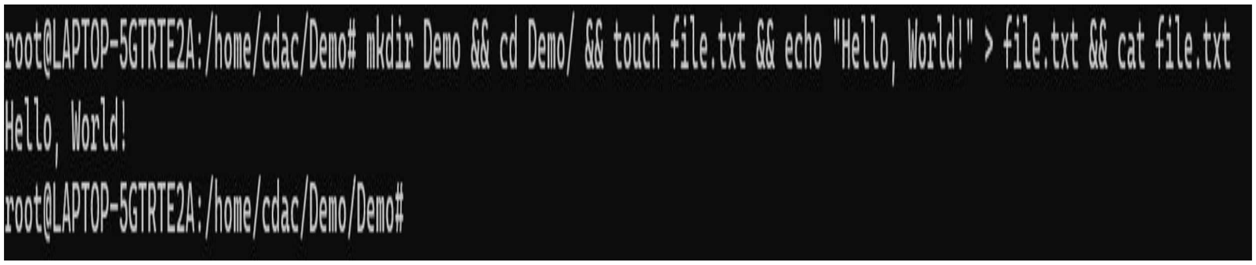
This command is used to terminate the running process in the system.



```
root@LAPTOP-5GTRTE2A:/home/cdac# kill 1
```

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

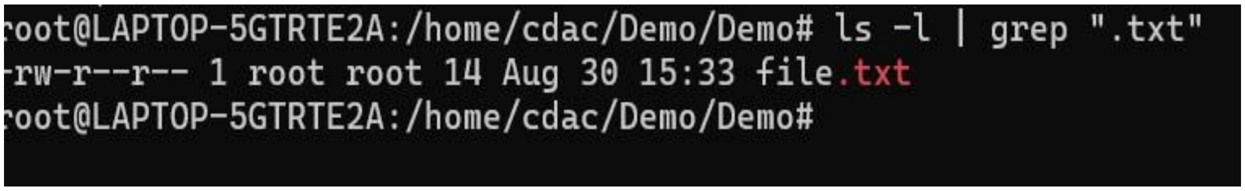
This commands creates a directory first then creates a file next writes a message to the file and then displays the message i.e Hello, World!.



```
root@LAPTOP-5GTRTE2A:/home/cdac/Demo# mkdir Demo && cd Demo/ && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt
Hello, World!
root@LAPTOP-5GTRTE2A:/home/cdac/Demo/Demo#
```

• ls -l | grep ".txt"

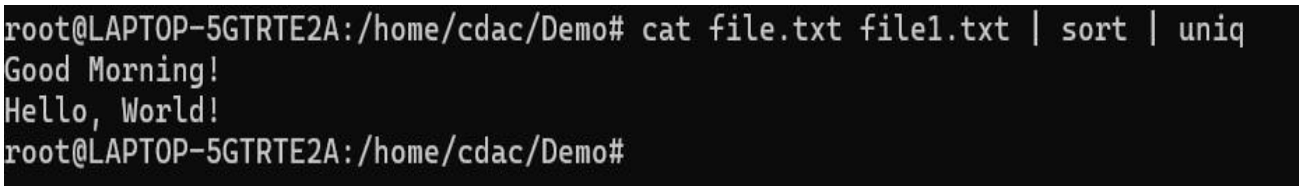
This command is use to combines two commands using a pipe (|) where the output of the first command “ ls -l ” is passed as input to the second command i.e. grep ".txt"



```
oot@LAPTOP-5GTRTE2A:/home/cdac/Demo/Demo# ls -l | grep ".txt"
-rw-r--r-- 1 root root 14 Aug 30 15:33 file.txt
oot@LAPTOP-5GTRTE2A:/home/cdac/Demo/Demo#
```

• cat file1.txt file2.txt | sort | uniq

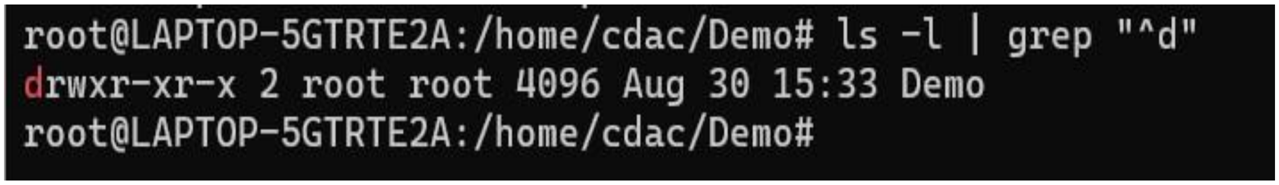
This command is used to show the unique lines in a sorted order.



```
root@LAPTOP-5GTRTE2A:/home/cdac/Demo# cat file.txt file1.txt | sort | uniq
Good Morning!
Hello, World!
root@LAPTOP-5GTRTE2A:/home/cdac/Demo#
```

• ls -l | grep "^d"

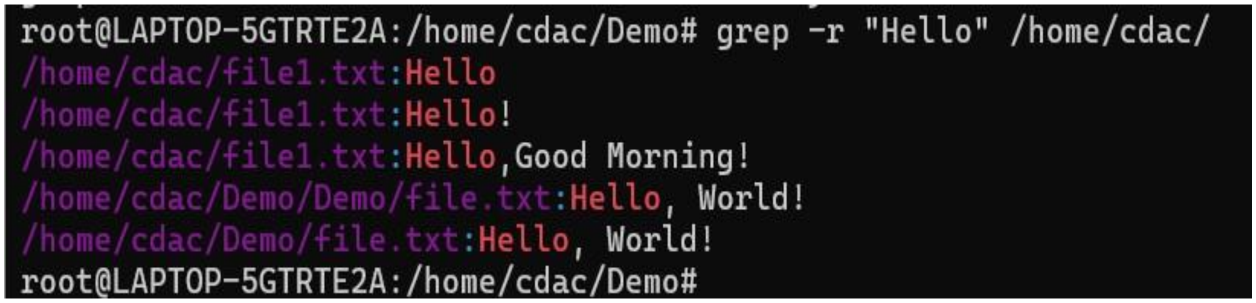
This command is used to list only the directories in the current directory.



```
root@LAPTOP-5GTRTE2A:/home/cdac/Demo# ls -l | grep "^d"
drwxr-xr-x 2 root root 4096 Aug 30 15:33 Demo
root@LAPTOP-5GTRTE2A:/home/cdac/Demo#
```

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

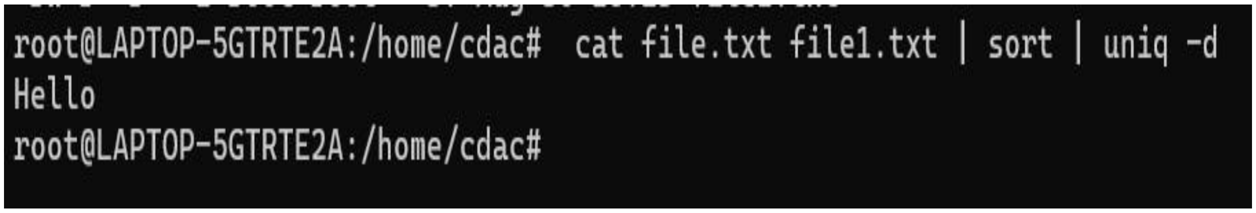
This command is used to search recursively for a specified pattern within all files in a given directory and its subdirectories.



```
root@LAPTOP-5GTRTE2A:/home/cdac/Demo# grep -r "Hello" /home/cdac/
/home/cdac/file1.txt:Hello
/home/cdac/file1.txt:Hello!
/home/cdac/file1.txt:Hello,Good Morning!
/home/cdac/Demo/Demo/file.txt:Hello, World!
/home/cdac/Demo/file.txt:Hello, World!
root@LAPTOP-5GTRTE2A:/home/cdac/Demo#
```

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

This command is used to find and display duplicate lines that appear in both files.



```
root@LAPTOP-5GTRTE2A:/home/cdac#  cat file.txt file1.txt | sort | uniq -d
Hello
root@LAPTOP-5GTRTE2A:/home/cdac#
```

• chmod 644 file.txt

This command is used to change the permission of a file. Here 644 is a numbers to specify the permission.

Read = 4

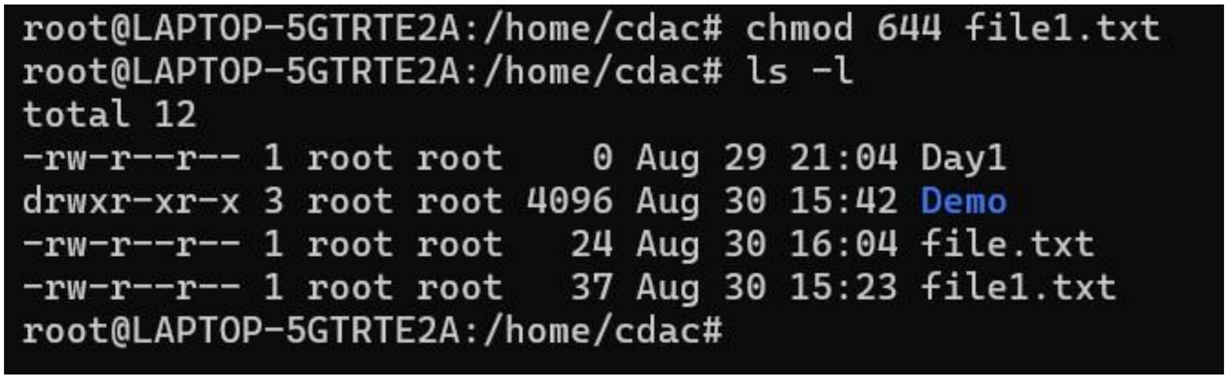
Write = 2

Execute = 1

6 => This number is for owner. 4+2 = 6 it means ower have permission to read & write.

4 => This number is for group. 4 which means group have permission for read only.

4 => This number is for all other. 4 which means others have permission for read only.



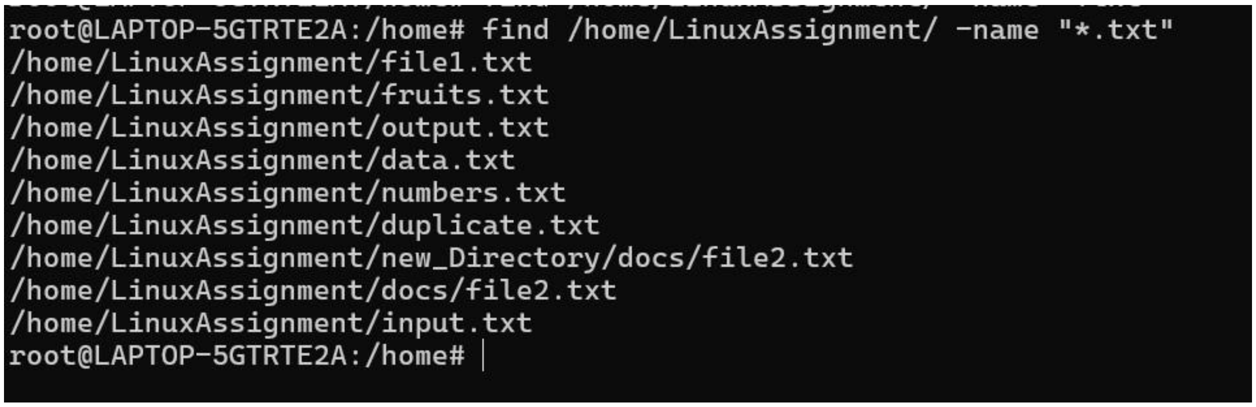
```
root@LAPTOP-5GTRTE2A:/home/cdac# chmod 644 file1.txt
root@LAPTOP-5GTRTE2A:/home/cdac# ls -l
total 12
-rw-r--r-- 1 root root    0 Aug 29 21:04 Day1
drwxr-xr-x 3 root root 4096 Aug 30 15:42 Demo
-rw-r--r-- 1 root root   24 Aug 30 16:04 file.txt
-rw-r--r-- 1 root root   37 Aug 30 15:23 file1.txt
root@LAPTOP-5GTRTE2A:/home/cdac#
```

• cp -r source_directory destination_directory

This command is used to copy a directory and all of its contents, including subdirectories, from one location to another.

• find /path/to/search -name "*.txt"

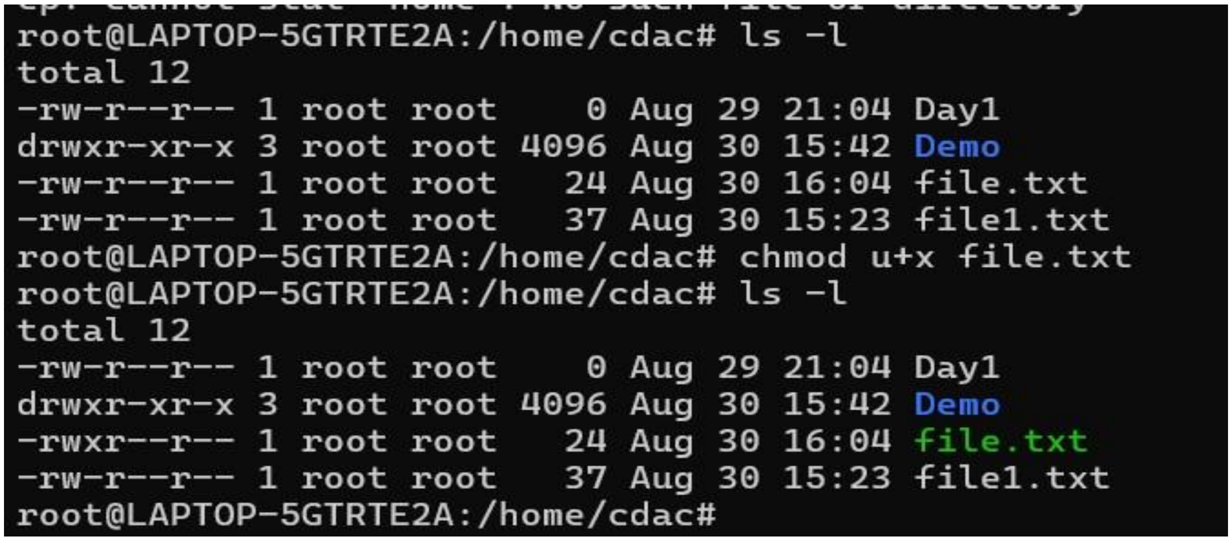
This is used to find files with a .txt extension within a specified directory and its subdirectories.



```
root@LAPTOP-5GTRTE2A:/home# find /home/LinuxAssignment/ -name "*.txt"
/home/LinuxAssignment/file1.txt
/home/LinuxAssignment/fruits.txt
/home/LinuxAssignment/output.txt
/home/LinuxAssignment/data.txt
/home/LinuxAssignment/numbers.txt
/home/LinuxAssignment/duplicate.txt
/home/LinuxAssignment/new_Directory/docs/file2.txt
/home/LinuxAssignment/docs/file2.txt
/home/LinuxAssignment/input.txt
root@LAPTOP-5GTRTE2A:/home#
```

• chmod u+x file.txt

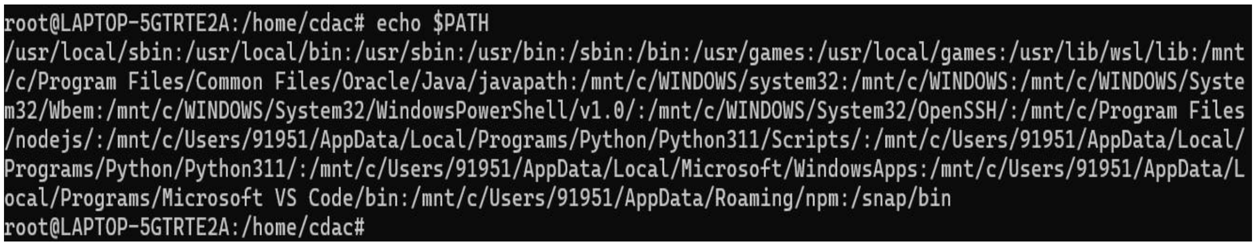
This is used to change the permissions of a file. Here we give permission to owner to execute the file.



```
root@LAPTOP-5GTRTE2A:/home/cdac# ls -l
total 12
-rw-r--r-- 1 root root    0 Aug 29 21:04 Day1
drwxr-xr-x 3 root root 4096 Aug 30 15:42 Demo
-rw-r--r-- 1 root root   24 Aug 30 16:04 file.txt
-rw-r--r-- 1 root root   37 Aug 30 15:23 file1.txt
root@LAPTOP-5GTRTE2A:/home/cdac# chmod u+x file.txt
root@LAPTOP-5GTRTE2A:/home/cdac# ls -l
total 12
-rw-r--r-- 1 root root    0 Aug 29 21:04 Day1
drwxr-xr-x 3 root root 4096 Aug 30 15:42 Demo
-rwxr--r-- 1 root root   24 Aug 30 16:04 file.txt
-rw-r--r-- 1 root root   37 Aug 30 15:23 file1.txt
root@LAPTOP-5GTRTE2A:/home/cdac#
```

• echo $PATH

This command is used to displays the current value of the PATH environment variable. The PATH variable contains a list of directories that the shell searches through when you type a command.



```
root@LAPTOP-5GTRTE2A:/home/cdac# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/lib/wsl/lib:/mnt
/c/Program Files/Common Files/Oracle/Java/javapath:/mnt/c/WINDOWS/system32:/mnt/c/WINDOWS:/mnt/c/WINDOWS/Syste
m32/Wbem:/mnt/c/WINDOWS/System32/WindowsPowerShell/v1.0/:/mnt/c/WINDOWS/System32/OpenSSH/:/mnt/c/Program Files
/nodejs/:/mnt/c/Users/91951/AppData/Local/Programs/Python/Python311/Scripts/:/mnt/c/Users/91951/AppData/Local/
Programs/Python/Python311/:/mnt/c/Users/91951/AppData/Local/Microsoft/WindowsApps:/mnt/c/Users/91951/AppData/L
ocal/Programs/Microsoft VS Code/bin:/mnt/c/Users/91951/AppData/Roaming/npm:/snap/bin
root@LAPTOP-5GTRTE2A:/home/cdac#
```

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**

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.

chmodx is the incorrecf command. The correct command is “chmod” which is use to change the permission.

2. **cpy** is used to copy files and directories

cpy is the incorrect command. The correct command is “cp” which is used to copy the file & directories.

3. **mkfile** is used to create a new file.

mkfile is the incorrect command. The correct command to create a new file is “touch”.

4. **catx** is used to concatenate files.

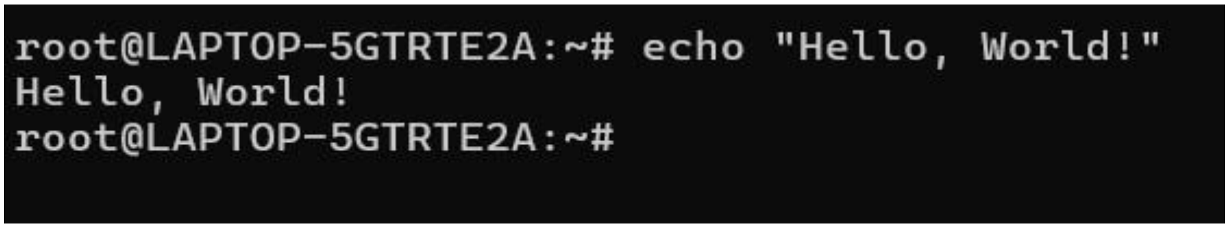
catx is incorrect command. The correct command is “cat” which is used to concatenate and display files.

5. **rn** is used to rename files.

rn is the incorrect command. The correct command is ‘’rm” which is used to remove a file.

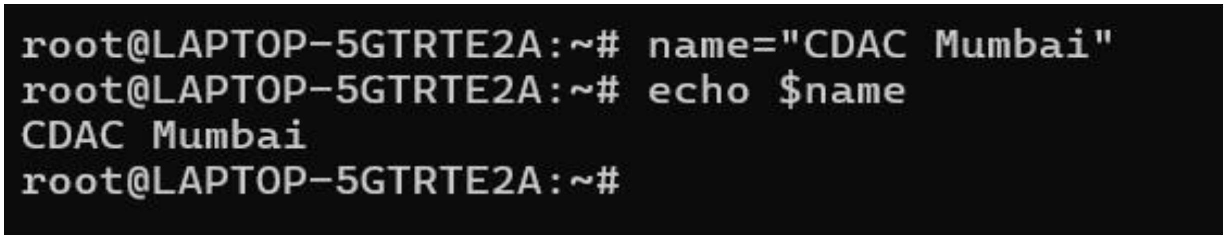
Part C

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



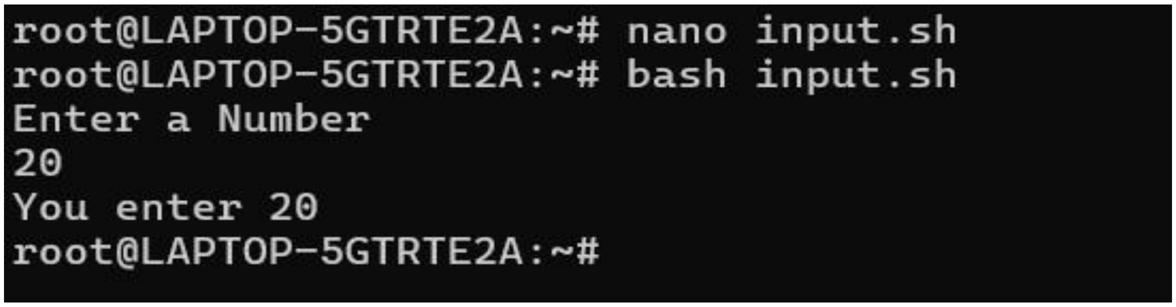
```
root@LAPTOP-5GTRTE2A:~# echo "Hello, World!"
Hello, World!
root@LAPTOP-5GTRTE2A:~#
```

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



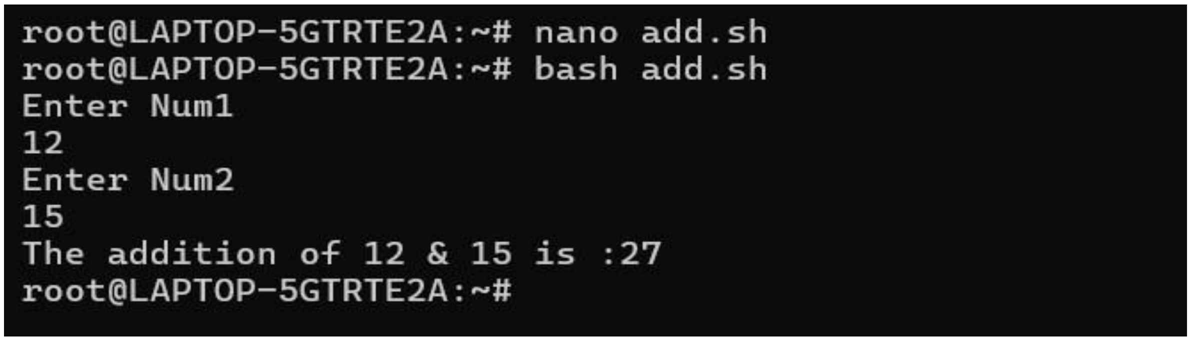
```
root@LAPTOP-5GTRTE2A:~# name="CDAC Mumbai"
root@LAPTOP-5GTRTE2A:~# echo $name
CDAC Mumbai
root@LAPTOP-5GTRTE2A:~#
```

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



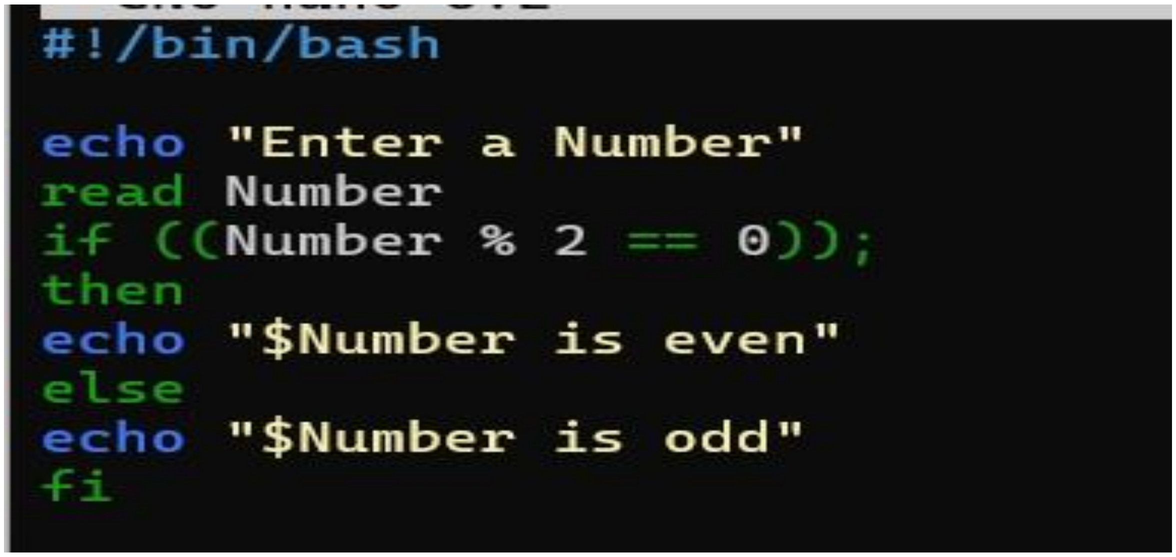
```
root@LAPTOP-5GTRTE2A:~# nano input.sh
root@LAPTOP-5GTRTE2A:~# bash input.sh
Enter a Number
20
You enter 20
root@LAPTOP-5GTRTE2A:~#
```

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

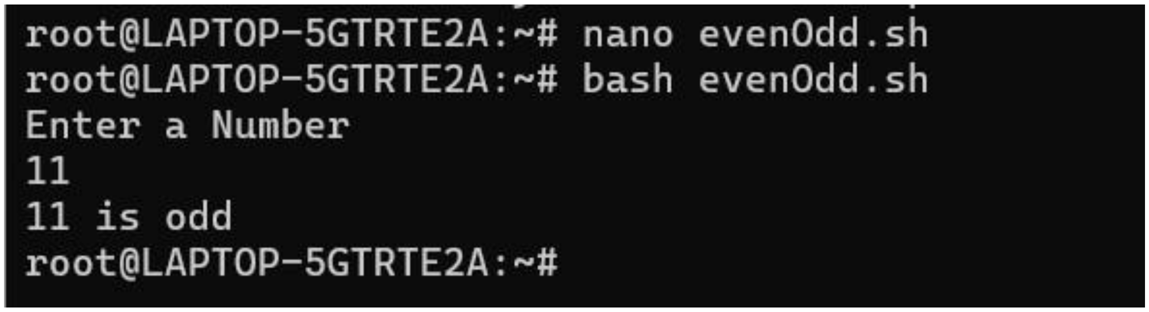


```
root@LAPTOP-5GTRTE2A:~# nano add.sh
root@LAPTOP-5GTRTE2A:~# bash add.sh
Enter Num1
12
Enter Num2
15
The addition of 12 & 15 is :27
root@LAPTOP-5GTRTE2A:~#
```

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

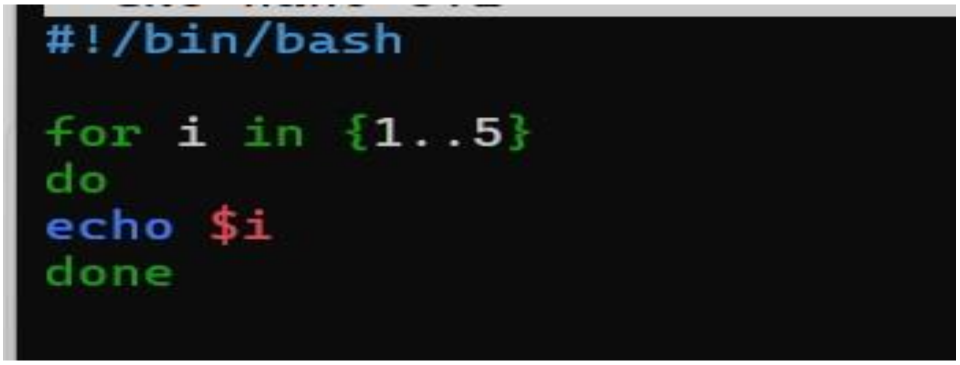


```
#!/bin/bash

echo "Enter a Number"
read Number
if ((Number % 2 == 0));
then
echo "$Number is even"
else
echo "$Number is odd"
fi
```

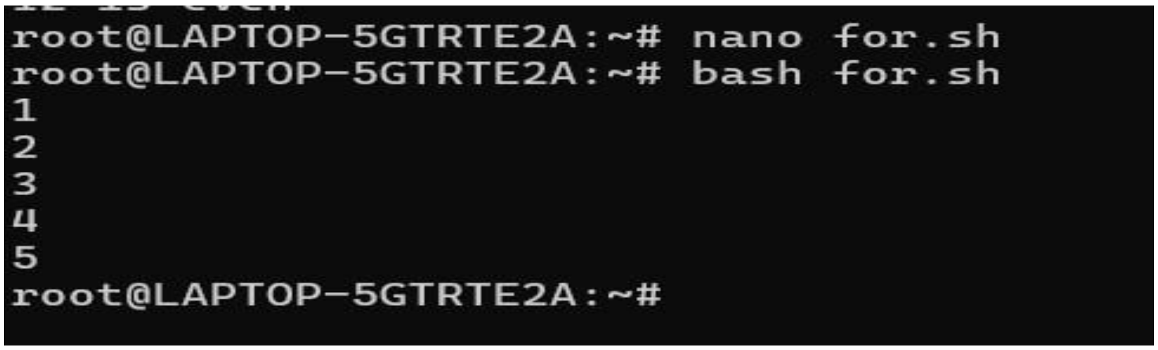


```
root@LAPTOP-5GTRTE2A:~# nano evenOdd.sh
root@LAPTOP-5GTRTE2A:~# bash evenOdd.sh
Enter a Number
11
11 is odd
root@LAPTOP-5GTRTE2A:~#
```

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

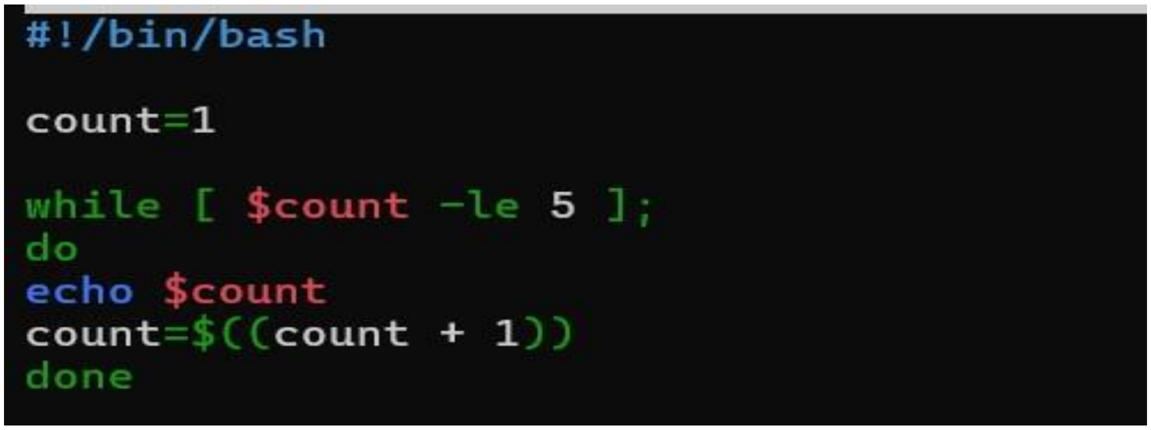


```
#!/bin/bash

for i in {1..5}
do
echo $i
done
```

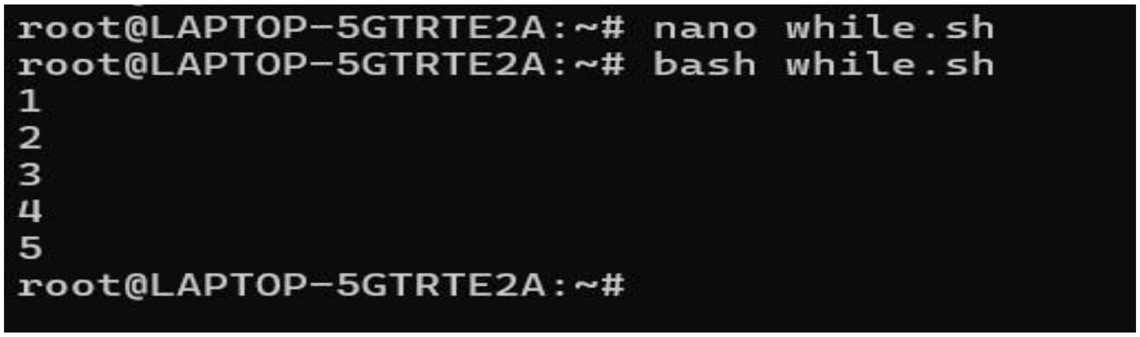


```
root@LAPTOP-5GTRTE2A:~# nano for.sh
root@LAPTOP-5GTRTE2A:~# bash for.sh
1
2
3
4
5
root@LAPTOP-5GTRTE2A:~#
```

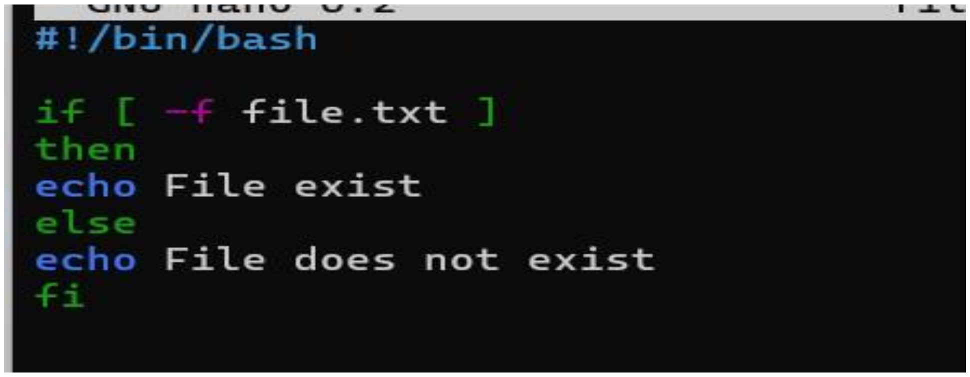
Question 7: Write a shell script that uses a while loop to print numbers from 1 to 5.

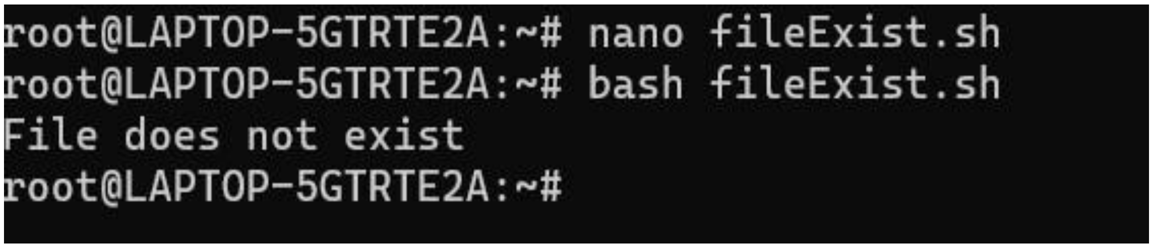


```
#!/bin/bash

count=1

while [ $count -le 5 ];
do
echo $count
count=$((count + 1))
done
```

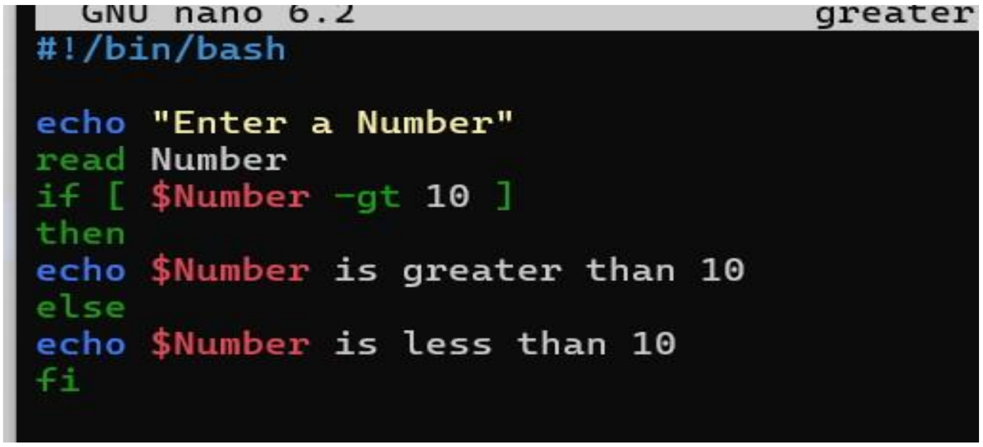

```
root@LAPTOP-5GTRTE2A:~# nano while.sh
root@LAPTOP-5GTRTE2A:~# bash while.sh
1
2
3
4
5
root@LAPTOP-5GTRTE2A:~#
```

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".

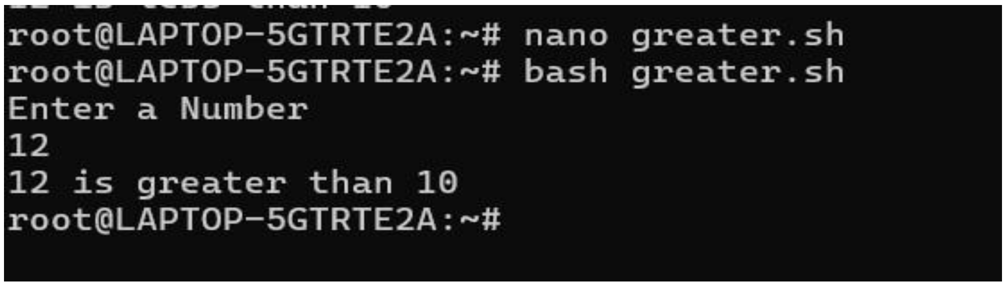

```
#!/bin/bash

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


```
root@LAPTOP-5GTRTE2A:~# nano fileExist.sh
root@LAPTOP-5GTRTE2A:~# bash fileExist.sh
File does not exist
root@LAPTOP-5GTRTE2A:~#
```

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.

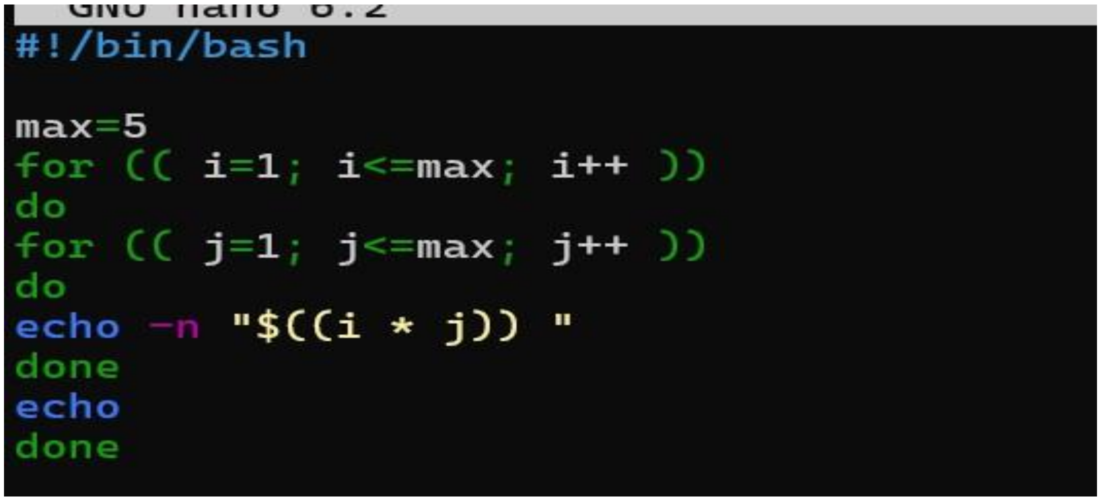


```
  GNU nano 6.2                     greater
#!/bin/bash

echo "Enter a Number"
read Number
if [ $Number -gt 10 ]
then
echo $Number is greater than 10
else
echo $Number is less than 10
fi
```

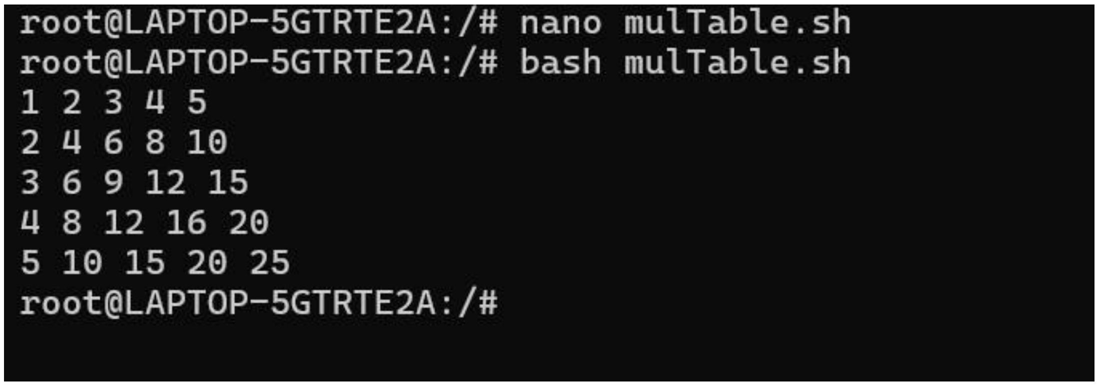


```
root@LAPTOP-5GTRTE2A:~# nano greater.sh
root@LAPTOP-5GTRTE2A:~# bash greater.sh
Enter a Number
12
12 is greater than 10
root@LAPTOP-5GTRTE2A:~#
```

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.

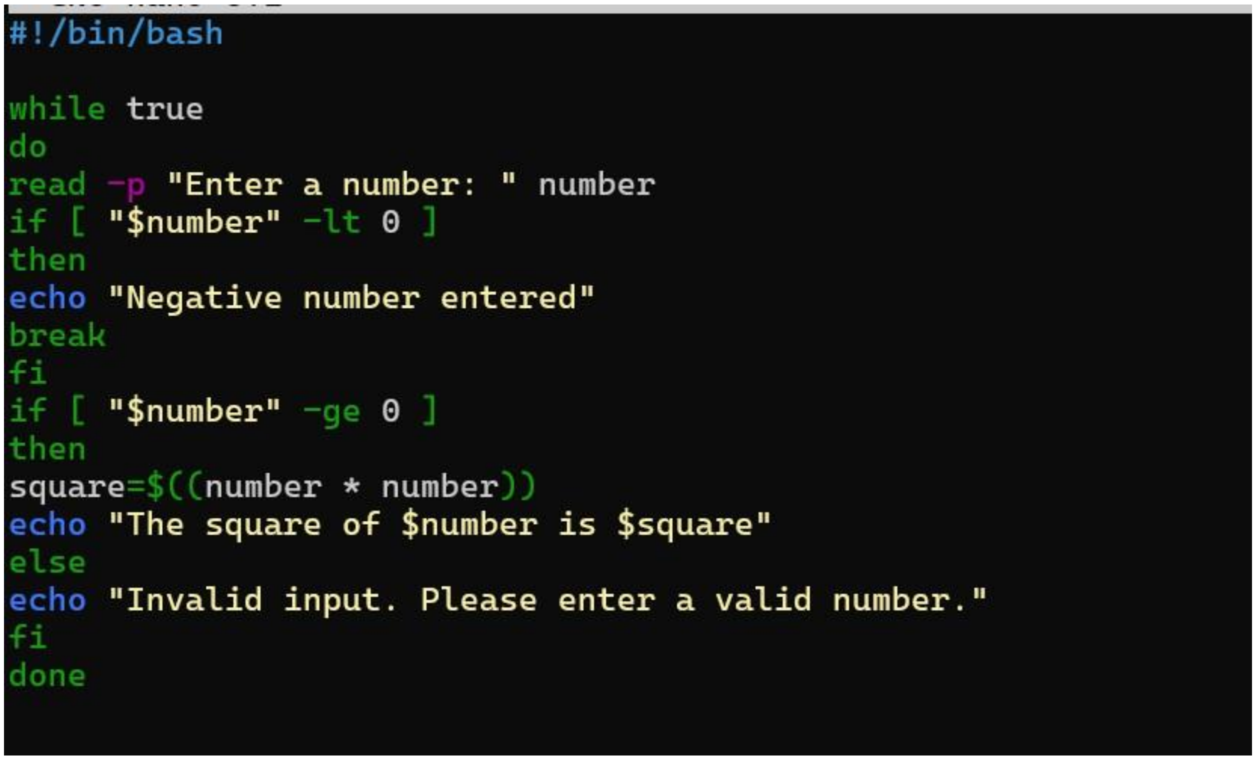


```
  GNU nano 6.2
#!/bin/bash

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

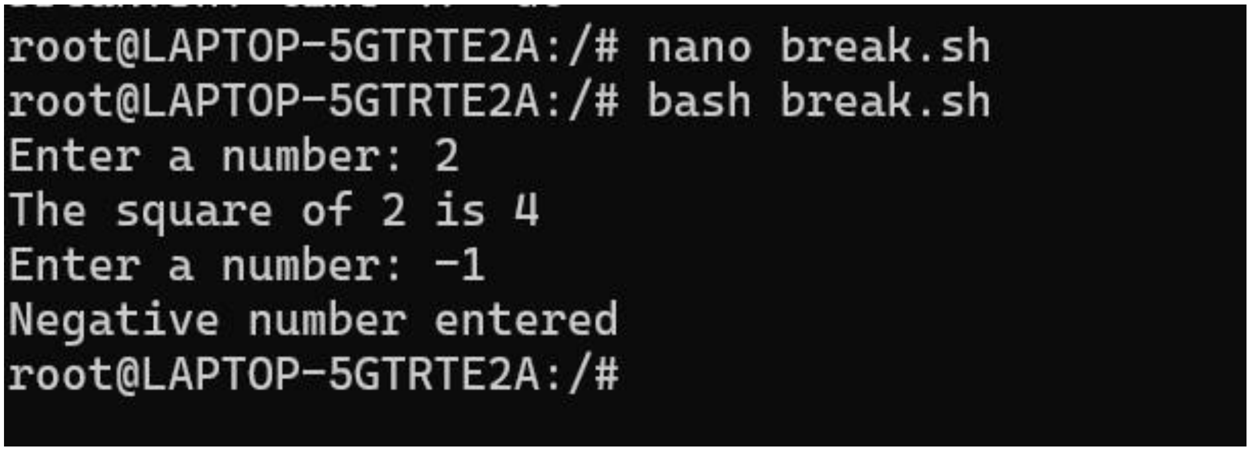


```
root@LAPTOP-5GTRTE2A:/# nano mulTable.sh
root@LAPTOP-5GTRTE2A:/# bash mulTable.sh
1 2 3 4 5
2 4 6 8 10
3 6 9 12 15
4 8 12 16 20
5 10 15 20 25
root@LAPTOP-5GTRTE2A:/#
```

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.



```
#!/bin/bash

while true
do
read -p "Enter a number: " number
if [ "$number" -lt 0 ]
then
echo "Negative number entered"
break
fi
if [ "$number" -ge 0 ]
then
square=$((number * number))
echo "The square of $number is $square"
else
echo "Invalid input. Please enter a valid number."
fi
done
```



```
root@LAPTOP-5GTRTE2A:/# nano break.sh
root@LAPTOP-5GTRTE2A:/# bash break.sh
Enter a number: 2
The square of 2 is 4
Enter a number: -1
Negative number entered
root@LAPTOP-5GTRTE2A:/#
```

Part E

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

| Process | Arrival Time | Burst Time |

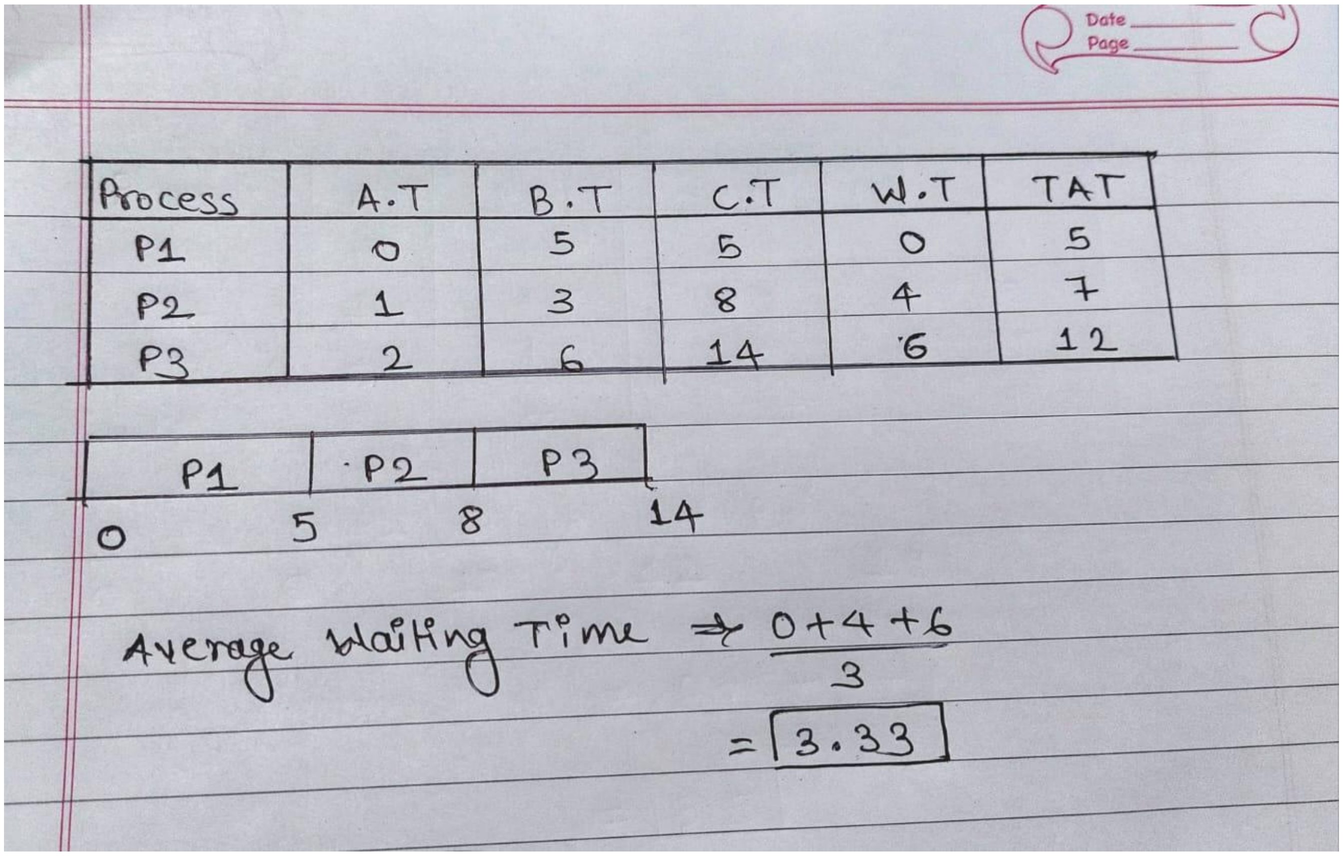
|---------|--------------|------------|

| P1 | 0 | 5 |

| P2 | 1 | 3 |

| P3 | 2 | 6 |

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



Process	A.T	B.T	C.T	W.T	TAT
P1	0	5	5	0	5
P2	1	3	8	4	7
P3	2	6	14	6	12

P1	P2	P3

0 5 8 14

Average Waiting Time $\Rightarrow \frac{0+4+6}{3}$

$= \boxed{3.33}$

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

| Process | Arrival Time | Burst Time |

|---------|--------------|------------|

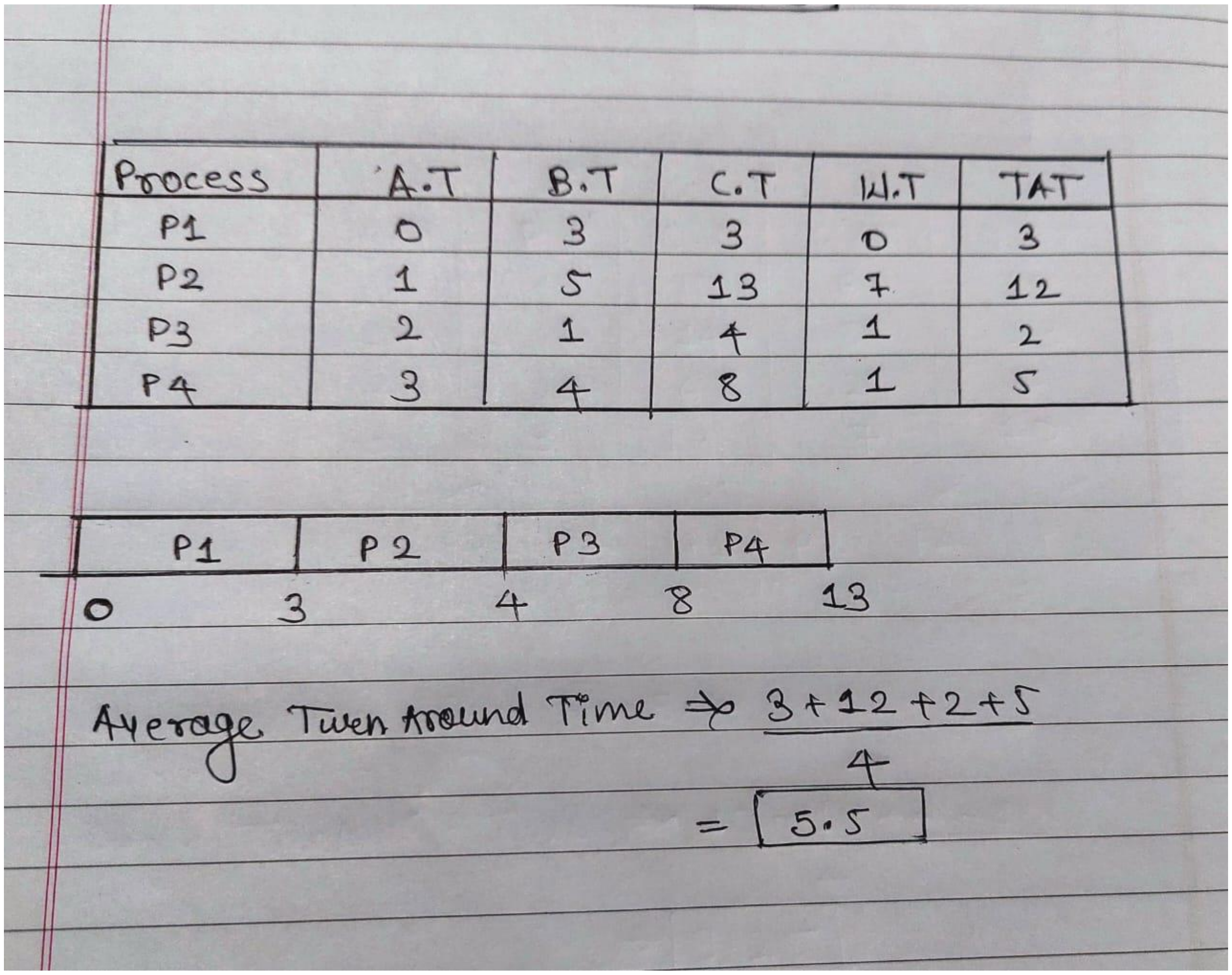
| P1 | 0 | 3 |

| P2 | 1 | 5 |

| P3 | 2 | 1 |

| P4 | 3 | 4 |

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



Process	A.T	B.T	C.T	W.T	TAT
P1	0	3	3	0	3
P2	1	5	13	7	12
P3	2	1	4	1	2
P4	3	4	8	1	5

P1
P2
P3
P4
0
3
4
8
13

Average Turn Around Time $\Rightarrow \frac{3+12+2+5}{4}$

$= \boxed{5.5}$

3. Consider the following processes with arrival times, burst times, and priorities (lower number indicates higher priority):

| Process | Arrival Time | Burst Time | Priority |

|---------|--------------|------------|----------|

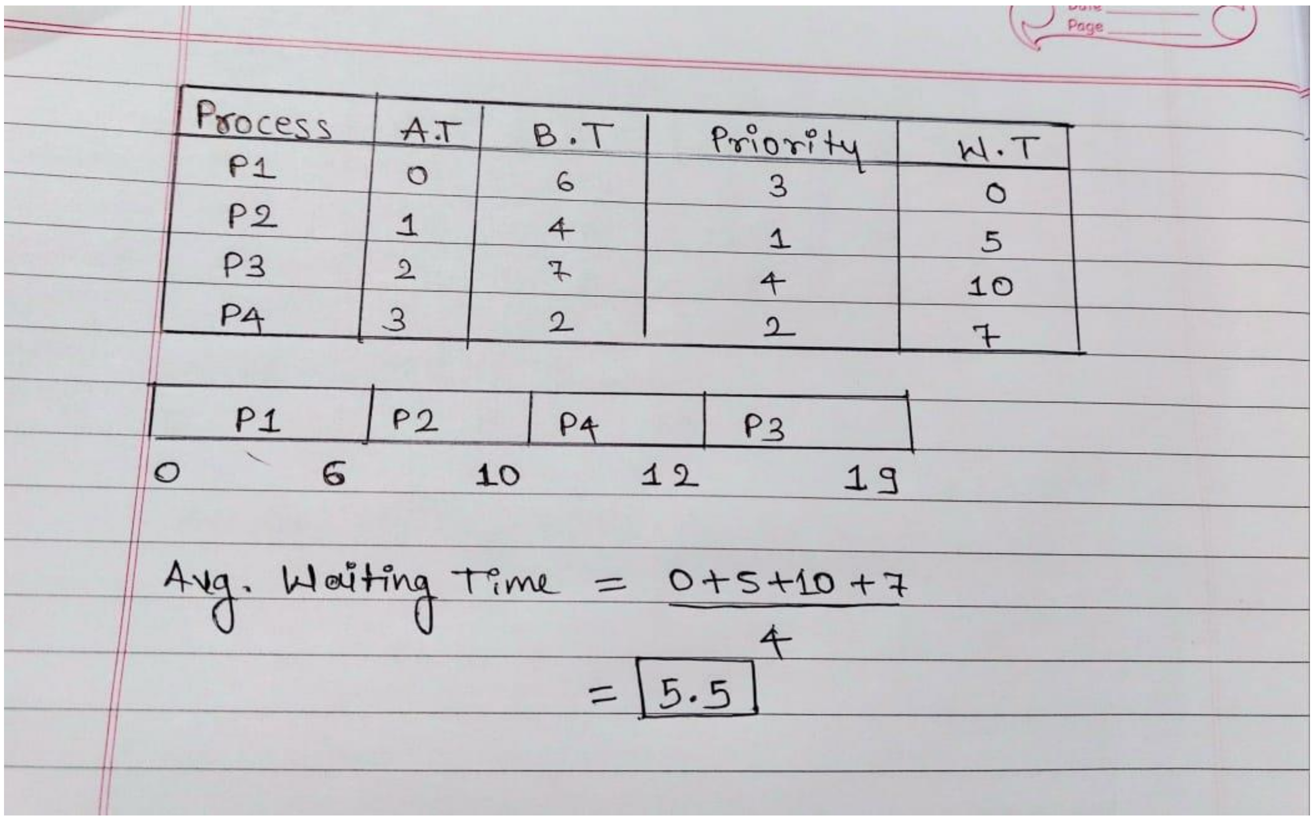
| P1 | 0 | 6 | 3 |

| P2 | 1 | 4 | 1 |

| P3 | 2 | 7 | 4 |

| P4 | 3 | 2 | 2 |

Calculate the average waiting time using Priority Scheduling.



Process	A.T	B.T	Priority	W.T
P1	0	6	3	0
P2	1	4	1	5
P3	2	7	4	10
P4	3	2	2	7

P1	P2	P4	P3

0 6 10 12 19

$$\text{Avg. Waiting Time} = \frac{0+5+10+7}{4}$$

$$= \boxed{5.5}$$

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 |

|---------|--------------|------------|

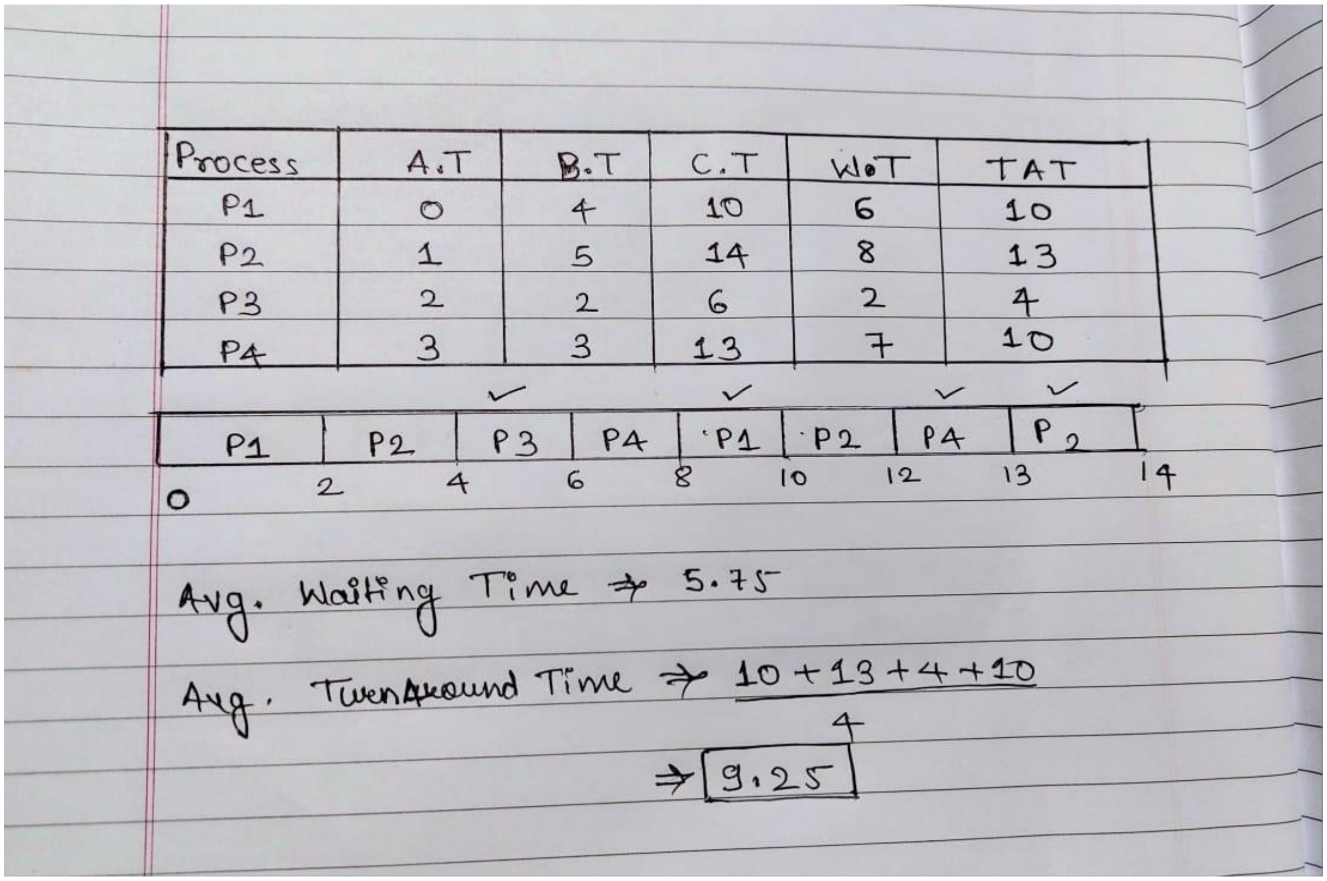
| P1 | 0 | 4 |

| P2 | 1 | 5 |

| P3 | 2 | 2 |

| P4 | 3 | 3 |

Calculate the average turnaround time using Round Robin scheduling.



Process	A.T	B.T	C.T	W.T	TAT
P1	0	4	10	6	10
P2	1	5	14	8	13
P3	2	2	6	2	4
P4	3	3	13	7	10

		✓		✓		✓	✓
P1	P2	P3	P4	P1	P2	P4	P2

0 2 4 6 8 10 12 13 14

Avg. Waiting Time ⇒ 5.75

Avg. TurnAround Time ⇒ $\frac{10+13+4+10}{4}$

⇒ $\boxed{9.25}$

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:-

Code:

```
#include <stdio.h>
void main()
{
int x=5;
fork();
x=x+1;
printf("%d\n",x);
}
```

The final values of **x** in the parent and child processes after the **fork()** call is **“6”**

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