CDAC MUMBAI

Concepts of Operating System

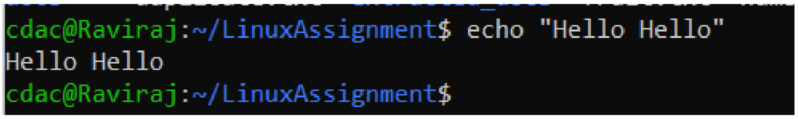
Assignment 2

Part A

What will the following commands do?

- echo "Hello, World!"

  Print the text "**Hello Hello**"

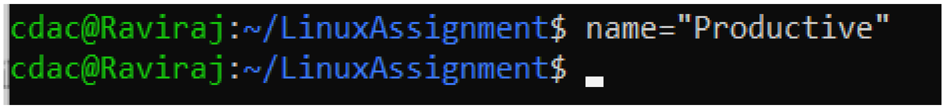


```
cdac@Raviraj:~/LinuxAssignment$ echo "Hello Hello"
Hello Hello
cdac@Raviraj:~/LinuxAssignment$
```

- name="Productive"

  assigns the string "Productive" to the variable name

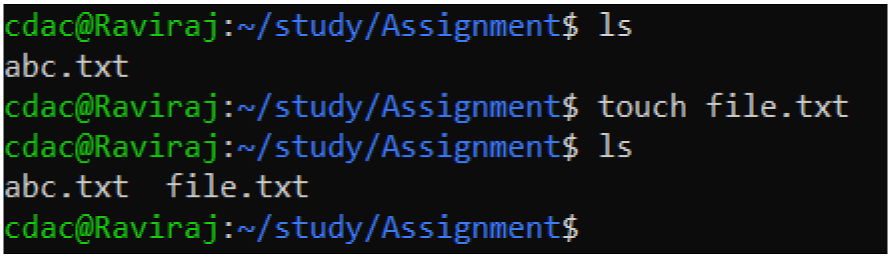
  The variable `name` can be used later in the script



```
cdac@Raviraj:~/LinuxAssignment$ name="Productive"
cdac@Raviraj:~/LinuxAssignment$
```

- touch file.txt

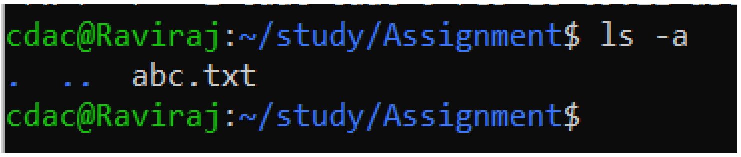
  If file.txt **does not exist**, this command creates an **empty** file named file.txt in the current directory.



```
cdac@Raviraj:~/study/Assignment$ ls
abc.txt
cdac@Raviraj:~/study/Assignment$ touch file.txt
cdac@Raviraj:~/study/Assignment$ ls
abc.txt  file.txt
cdac@Raviraj:~/study/Assignment$
```

- ls -a

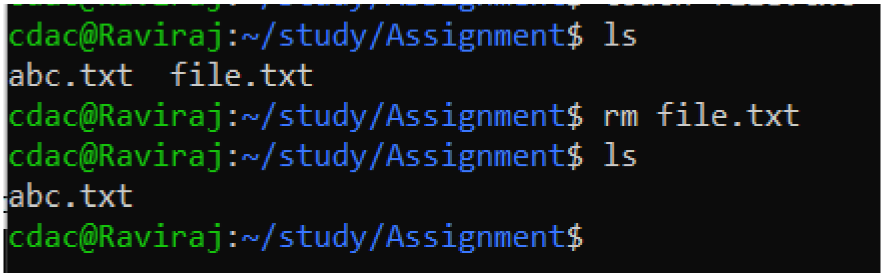
ls lists the files and directories in the current directory.
The -a option (short for **all**) includes **hidden files** (files that start with .).



```
cdac@Raviraj:~/study/Assignment$ ls -a
.  ..  abc.txt
cdac@Raviraj:~/study/Assignment$
```

- rm file.txt

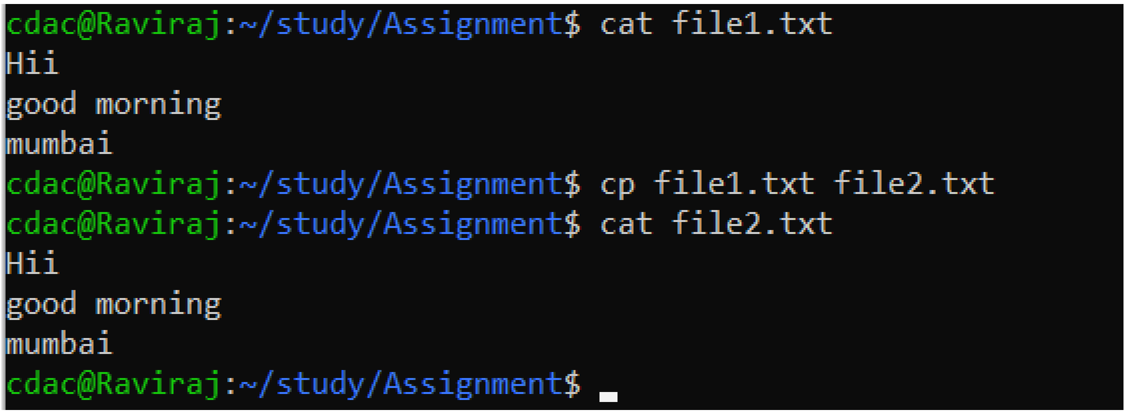
`rm` (remove) deletes the file named **file.txt** from the current directory.



```
cdac@Raviraj:~/study/Assignment$ ls
abc.txt  file.txt
cdac@Raviraj:~/study/Assignment$ rm file.txt
cdac@Raviraj:~/study/Assignment$ ls
abc.txt
cdac@Raviraj:~/study/Assignment$
```

- cp file1.txt file2.txt

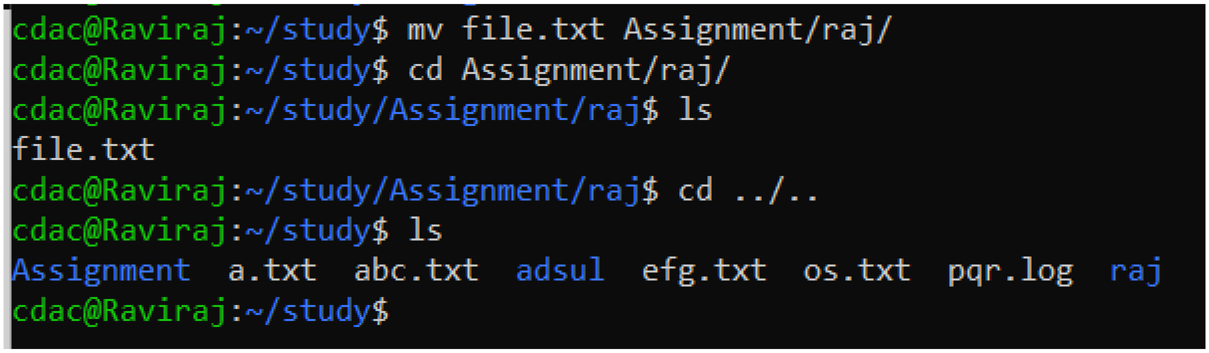
`cp` (copy) creates a duplicate of **file1.txt** and names it **file2.txt**.
cp t



```
cdac@Raviraj:~/study/Assignment$ cat file1.txt
Hii
good morning
mumbai
cdac@Raviraj:~/study/Assignment$ cp file1.txt file2.txt
cdac@Raviraj:~/study/Assignment$ cat file2.txt
Hii
good morning
mumbai
cdac@Raviraj:~/study/Assignment$
```

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

move `file.txt` from current directory to target directory



```
cdac@Raviraj:~/study$ mv file.txt Assignment/raj/
cdac@Raviraj:~/study$ cd Assignment/raj/
cdac@Raviraj:~/study/Assignment/raj$ ls
file.txt
cdac@Raviraj:~/study/Assignment/raj$ cd ../..
cdac@Raviraj:~/study$ ls
Assignment  a.txt  abc.txt  adsul  efg.txt  os.txt  pqr.log  raj
cdac@Raviraj:~/study$
```

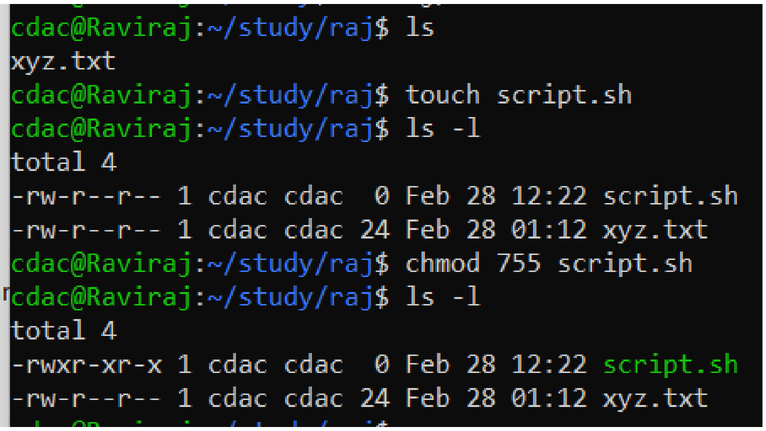
- chmod 755 script.sh

`chmod` (change mode) modifies file permissions

`755` sets specific permissions:

  - **7 (Owner)**: Read, write, and execute (rwx)
  - **5 (Group)**: Read and execute (`r-x`)
  - **5 (Others)**: Read and execute (`r-x`)

script.sh is the file whose permissions are being changed

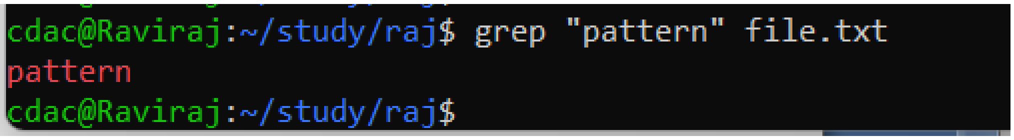


```
cdac@Raviraj:~/study/raj$ ls
xyz.txt
cdac@Raviraj:~/study/raj$ touch script.sh
cdac@Raviraj:~/study/raj$ ls -l
total 4
-rw-r--r-- 1 cdac cdac  0 Feb 28 12:22 script.sh
-rw-r--r-- 1 cdac cdac 24 Feb 28 01:12 xyz.txt
cdac@Raviraj:~/study/raj$ chmod 755 script.sh
cdac@Raviraj:~/study/raj$ ls -l
total 4
-rwxr-xr-x 1 cdac cdac  0 Feb 28 12:22 script.sh
-rw-r--r-- 1 cdac cdac 24 Feb 28 01:12 xyz.txt
```

- grep "pattern" file.txt

`grep` (Global Regular Expression Print) is used to search for a specific pattern in a file

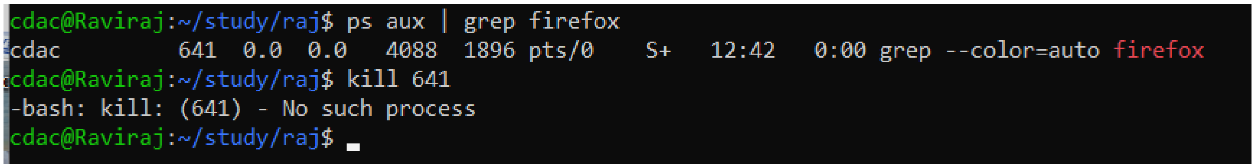
`"pattern"` is the text search into `file.txt` file



```
cdac@Raviraj:~/study/raj$ grep "pattern" file.txt
pattern
cdac@Raviraj:~/study/raj$
```

- kill PID

kill is used to **terminate a process**.
PID (Process ID) is the unique identifier assigned to a running process



```
cdac@Raviraj:~/study/raj$ ps aux | grep firefox
cdac         641  0.0  0.0   4088  1896 pts/0    S+   12:42   0:00 grep --color=auto firefox
cdac@Raviraj:~/study/raj$ kill 641
-bash: kill: (641) - No such process
cdac@Raviraj:~/study/raj$ 
```

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

This command performs multiple actions **sequentially**, using && to ensure each step succeeds before proceeding.

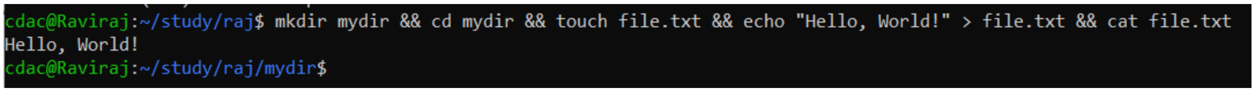
mkdir mydir
  - Creates a new directory named **mydir**.

cd mydir
  - Changes the current directory to **mydir**.

touch file.txt
  - Creates an empty file named **file.txt**.

echo "Hello, World!" > file.txt
  - Writes "Hello, World!" into file.txt, overwriting any existing content.

cat file.txt
  - Displays the contents of file.txt (which should be "Hello, World!").

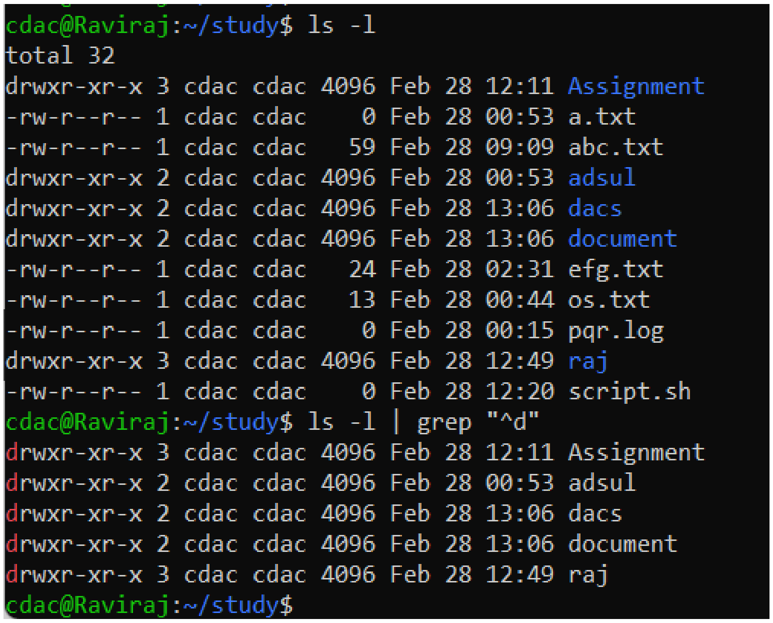


```
cdac@Raviraj:~/study/raj$ mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt
Hello, World!
cdac@Raviraj:~/study/raj/mydir$
```

- ls -l | grep "^d"

ls -l
  - Lists all files and directories in long format (permissions, owner, size, date, etc.).

grep "^d"
  - Filters lines that **start (^) with "d"**, which indicates directories in `ls -l` output.



```
cdac@Raviraj:~/study$ ls -l
total 32
drwxr-xr-x 3 cdac cdac 4096 Feb 28 12:11 Assignment
-rw-r--r-- 1 cdac cdac    0 Feb 28 00:53 a.txt
-rw-r--r-- 1 cdac cdac   59 Feb 28 09:09 abc.txt
drwxr-xr-x 2 cdac cdac 4096 Feb 28 00:53 adsul
drwxr-xr-x 2 cdac cdac 4096 Feb 28 13:06 dacs
drwxr-xr-x 2 cdac cdac 4096 Feb 28 13:06 document
-rw-r--r-- 1 cdac cdac   24 Feb 28 02:31 efg.txt
-rw-r--r-- 1 cdac cdac   13 Feb 28 00:44 os.txt
-rw-r--r-- 1 cdac cdac    0 Feb 28 00:15 pqr.log
drwxr-xr-x 3 cdac cdac 4096 Feb 28 12:49 raj
-rw-r--r-- 1 cdac cdac    0 Feb 28 12:20 script.sh
cdac@Raviraj:~/study$ ls -l | grep "^d"
drwxr-xr-x 3 cdac cdac 4096 Feb 28 12:11 Assignment
drwxr-xr-x 2 cdac cdac 4096 Feb 28 00:53 adsul
drwxr-xr-x 2 cdac cdac 4096 Feb 28 13:06 dacs
drwxr-xr-x 2 cdac cdac 4096 Feb 28 13:06 document
drwxr-xr-x 3 cdac cdac 4096 Feb 28 12:49 raj
cdac@Raviraj:~/study$
```

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

This command **searches for a specific pattern recursively** in all files inside a given directory.

grep

- A command used to search for text patterns in files

`-r` (Recursive)

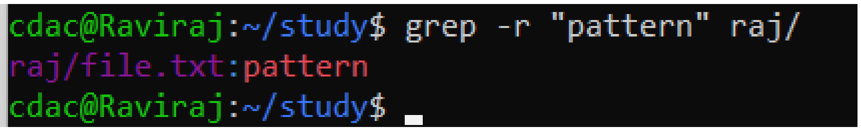
- Searches through **all files and subdirectories** inside `/path/to/directory/`

"pattern"

- The text or regular expression you are searching for.

/path/to/directory/

- Specifies the directory where the search should start.



```
cdac@Raviraj:~/study$ grep -r "pattern" raj/
raj/file.txt:pattern
cdac@Raviraj:~/study$
```

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

This command finds **duplicate lines** that appear in both `file1.txt` and `file2.txt`.

cat file1.txt file2.txt

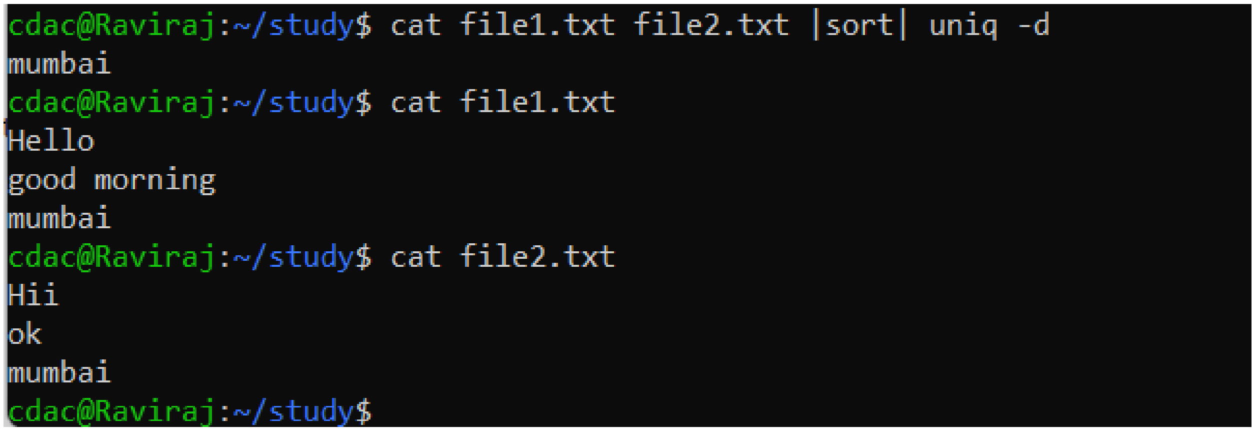
- Concatenates (merges) the contents of `file1.txt` and `file2.txt` and sends them to the next command.

sort

- Sorts the merged output (because `uniq` only works on sorted data)

uniq -d

- Extracts and prints **only duplicate lines** (lines that appear more than once).

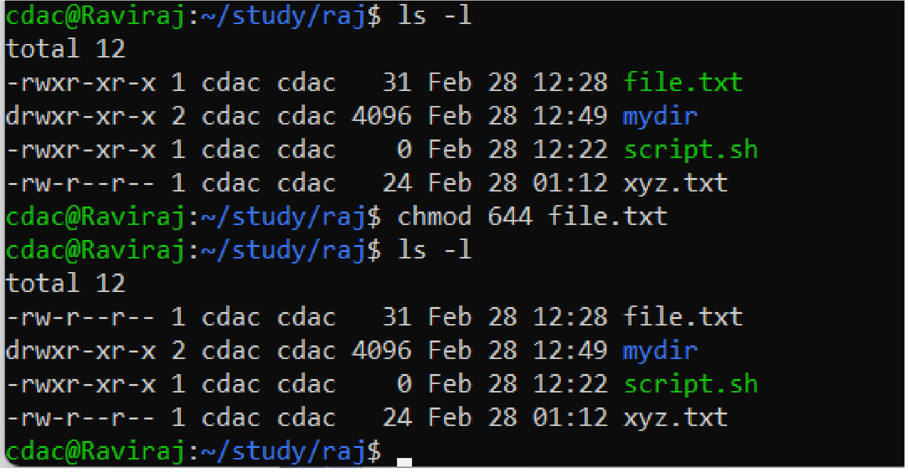


```
cdac@Raviraj:~/study$ cat file1.txt file2.txt |sort| uniq -d
mumbai
cdac@Raviraj:~/study$ cat file1.txt
Hello
good morning
mumbai
cdac@Raviraj:~/study$ cat file2.txt
Hii
ok
mumbai
cdac@Raviraj:~/study$
```

- chmod 644 file.txt

This command changes the file permissions of `file.txt` to **644**, meaning:

- **Owner**: Read (4) + Write (2) = **6**
- **Group**: Read (4) = **4**
- **Others**: Read (4) = **4**



```
cdac@Raviraj:~/study/raj$ ls -l
total 12
-rwxr-xr-x 1 cdac cdac   31 Feb 28 12:28 file.txt
drwxr-xr-x 2 cdac cdac 4096 Feb 28 12:49 mydir
-rwxr-xr-x 1 cdac cdac    0 Feb 28 12:22 script.sh
-rw-r--r-- 1 cdac cdac   24 Feb 28 01:12 xyz.txt
cdac@Raviraj:~/study/raj$ chmod 644 file.txt
cdac@Raviraj:~/study/raj$ ls -l
total 12
-rw-r--r-- 1 cdac cdac   31 Feb 28 12:28 file.txt
drwxr-xr-x 2 cdac cdac 4096 Feb 28 12:49 mydir
-rwxr-xr-x 1 cdac cdac    0 Feb 28 12:22 script.sh
-rw-r--r-- 1 cdac cdac   24 Feb 28 01:12 xyz.txt
cdac@Raviraj:~/study/raj$ 
```

- cp -r source_directory destination_directory

This command **copies a directory (source_directory) and its contents** to a new location (destination_directory)

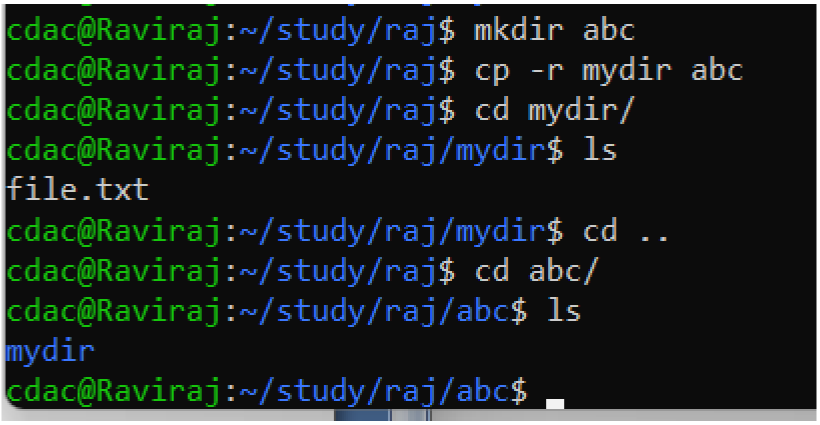
cp → Stands for "copy"

-r (or --recursive) → Enables recursive copying, meaning:

  - It copies all files **and subdirectories** inside source_directory

source_directory → The directory you want to copy.

destination_directory → The target location where the directory should be copied.



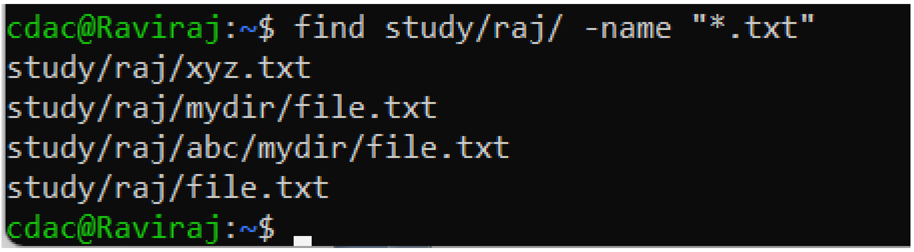
```
cdac@Raviraj:~/study/raj$ mkdir abc
cdac@Raviraj:~/study/raj$ cp -r mydir abc
cdac@Raviraj:~/study/raj$ cd mydir/
cdac@Raviraj:~/study/raj/mydir$ ls
file.txt
cdac@Raviraj:~/study/raj/mydir$ cd ..
cdac@Raviraj:~/study/raj$ cd abc/
cdac@Raviraj:~/study/raj/abc$ ls
mydir
cdac@Raviraj:~/study/raj/abc$ _
```

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

find → used to search for files and directories.

/path/to/search → The directory where the search begins.

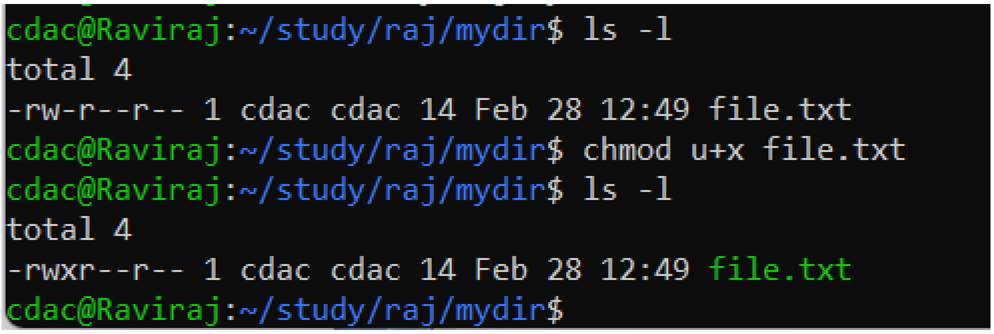
-name "*.txt" → Finds files with a .txt extension.



```
cdac@Raviraj:~$ find study/raj/ -name "*.txt"
study/raj/xyz.txt
study/raj/mydir/file.txt
study/raj/abc/mydir/file.txt
study/raj/file.txt
cdac@Raviraj:~$ _
```

- chmod u+x file.txt

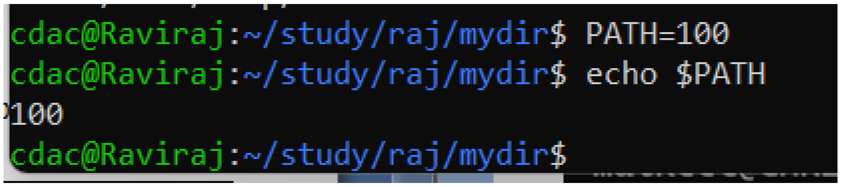
chmod → Changes file permissions.
u → Stands for "user" (the owner of the file).
+x → Adds execute (x) permission.
file.txt → The target file.



```
cdac@Raviraj:~/study/raj/mydir$ ls -l
total 4
-rw-r--r-- 1 cdac cdac 14 Feb 28 12:49 file.txt
cdac@Raviraj:~/study/raj/mydir$ chmod u+x file.txt
cdac@Raviraj:~/study/raj/mydir$ ls -l
total 4
-rwxr--r-- 1 cdac cdac 14 Feb 28 12:49 file.txt
cdac@Raviraj:~/study/raj/mydir$
```

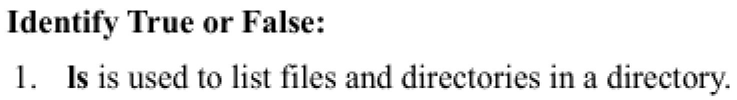
- echo $PATH

echo → Displays the value of a variable.
$PATH → Represents the environment variable that contains directories where executable files are searched



```
cdac@Raviraj:~/study/raj/mydir$ PATH=100
cdac@Raviraj:~/study/raj/mydir$ echo $PATH
100
cdac@Raviraj:~/study/raj/mydir$
```

Part B



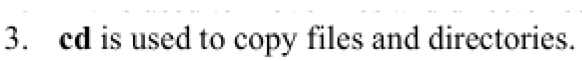
Identify True or False:

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

- **True** – `ls` lists files and directories in a directory

2. **mv** is used to move files and directories.

- **True** – `mv` moves files and directories.



3. **cd** is used to copy files and directories.

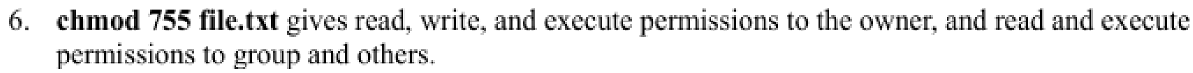
- **False** – `cd` is used to change directories, not copy files and directories.

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

- **True** – `pwd` prints the current working directory.

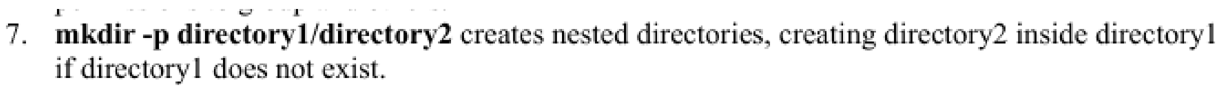
5. **grep** is used to search for patterns in files.

- **True** – `grep` searches for patterns in files



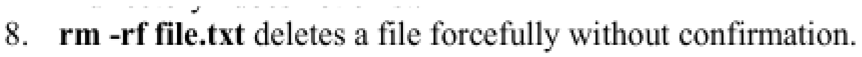
6. **chmod 755 file.txt** gives read, write, and execute permissions to the owner, and read and execute permissions to group and others.

- **True** – `chmod 755 file.txt` gives read, write, and execute permissions to the owner, and read/execute permissions to group and others.



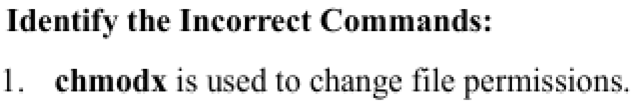
7. **mkdir -p directory1/directory2** creates nested directories, creating directory2 inside directory1 if directory1 does not exist.

- **True** – `mkdir -p directory1/directory2` creates nested directories.



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

- **True** – `rm -rf file.txt` deletes a file forcefully without confirmation.



Identify the Incorrect Commands:

1. **chmodx** is used to change file permissions.

- **Incorrect:** chmodx → **Correct:** chmod (used to change file permissions).

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

- **Incorrect:** cpy → **Correct:** cp (used to copy files and directories).

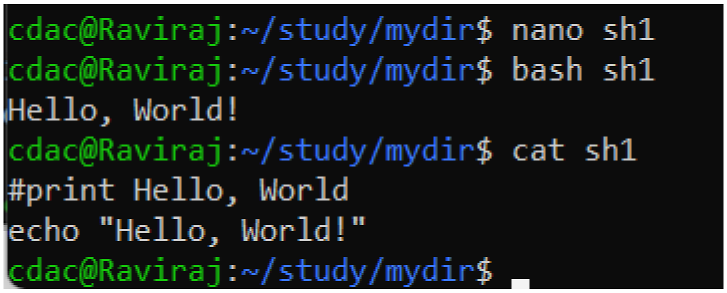
3. **mkfile** is used to create a new file.
   - **Incorrect:** mkfile → **Correct:** touch (used to create a new file).

4. **catx** is used to concatenate files.
   - **Incorrect:** catx → **Correct:** cat (used to concatenate files).

5. **rn** is used to rename files.
   - **Incorrect:** rn → **Correct:** mv (used to rename or move files)

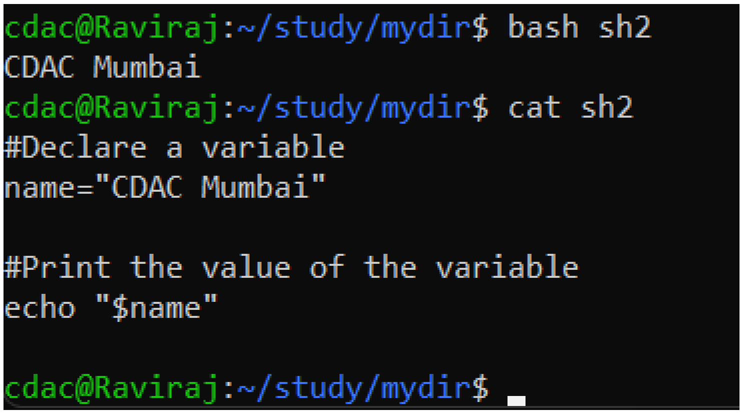
Part C

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



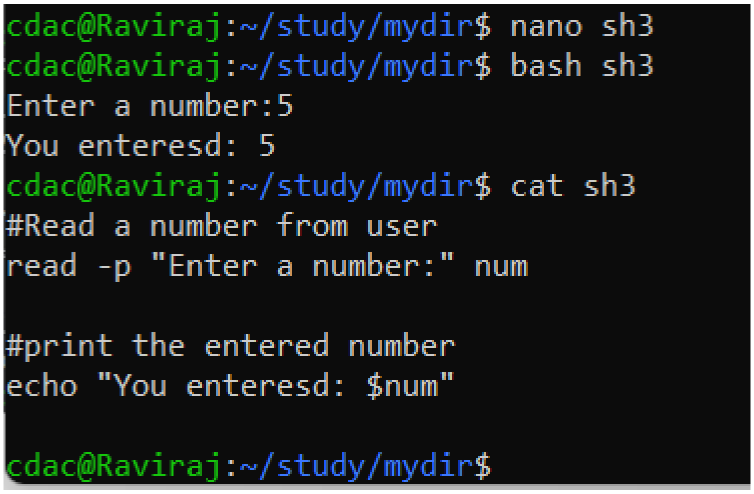
```
cdac@Raviraj:~/study/mydir$ nano sh1
cdac@Raviraj:~/study/mydir$ bash sh1
Hello, World!
cdac@Raviraj:~/study/mydir$ cat sh1
#print Hello, World
echo "Hello, World!"
cdac@Raviraj:~/study/mydir$
```

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



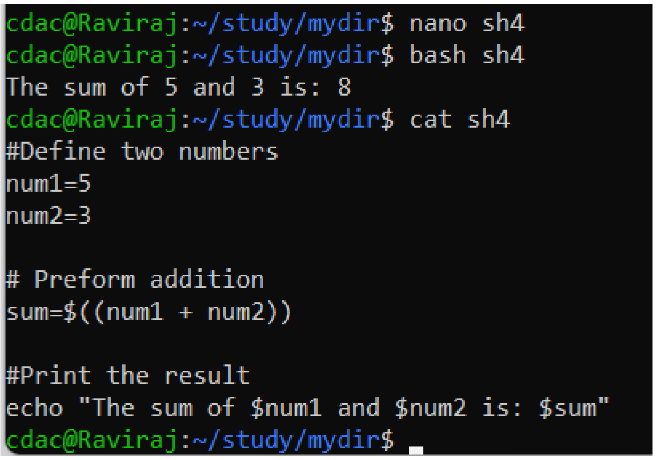
```
cdac@Raviraj:~/study/mydir$ bash sh2
CDAC Mumbai
cdac@Raviraj:~/study/mydir$ cat sh2
#Declare a variable
name="CDAC Mumbai"

#Print the value of the variable
echo "$name"

cdac@Raviraj:~/study/mydir$
```

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



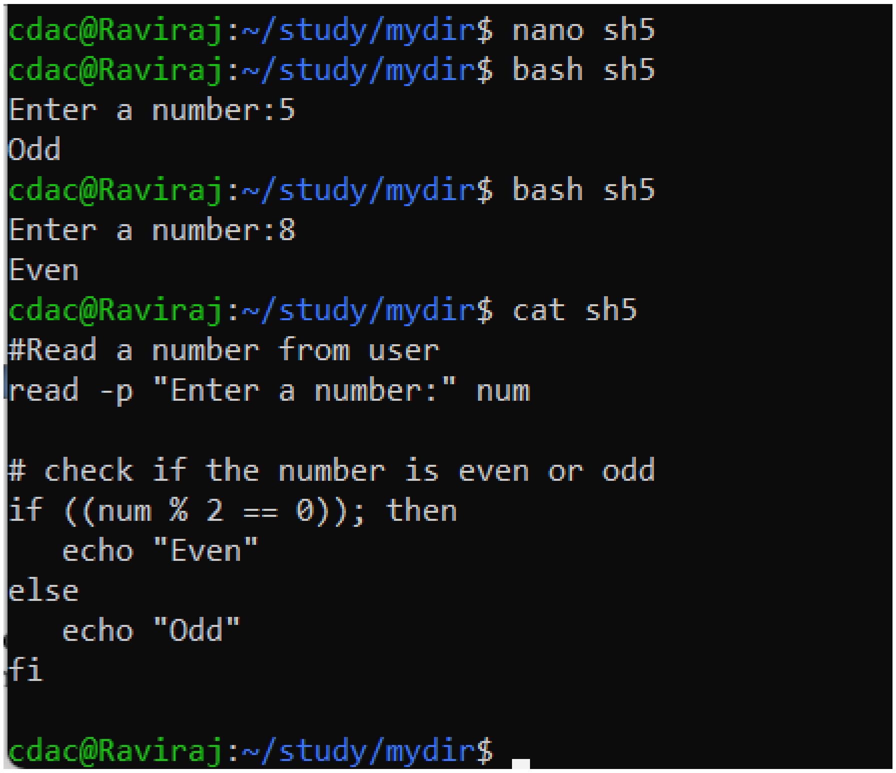
```
cdac@Raviraj:~/study/mydir$ nano sh3
cdac@Raviraj:~/study/mydir$ bash sh3
Enter a number:5
You enteresd: 5
cdac@Raviraj:~/study/mydir$ cat sh3
#Read a number from user
read -p "Enter a number:" num

#print the entered number
echo "You enteresd: $num"

cdac@Raviraj:~/study/mydir$
```

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



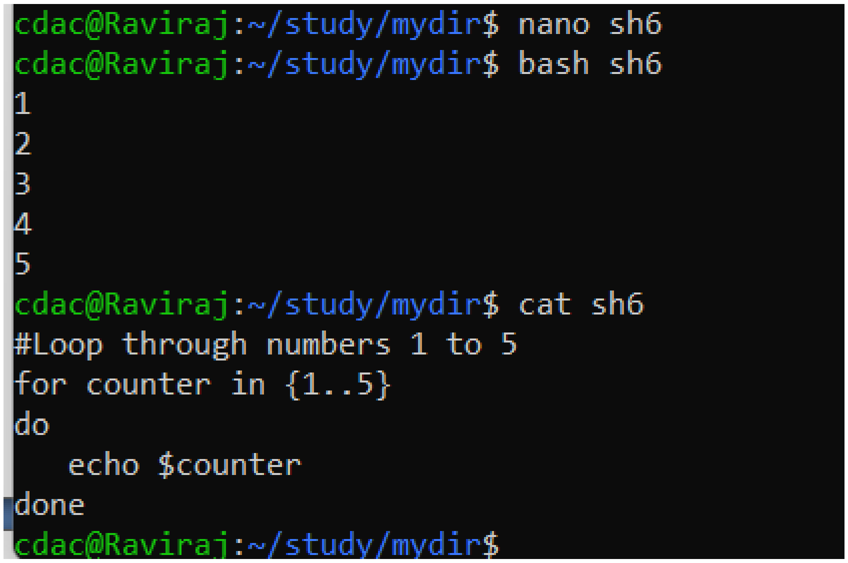
```
cdac@Raviraj:~/study/mydir$ nano sh4
cdac@Raviraj:~/study/mydir$ bash sh4
The sum of 5 and 3 is: 8
cdac@Raviraj:~/study/mydir$ cat sh4
#Define two numbers
num1=5
num2=3

# Preform addition
sum=$((num1 + num2))

#Print the result
echo "The sum of $num1 and $num2 is: $sum"
cdac@Raviraj:~/study/mydir$
```

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



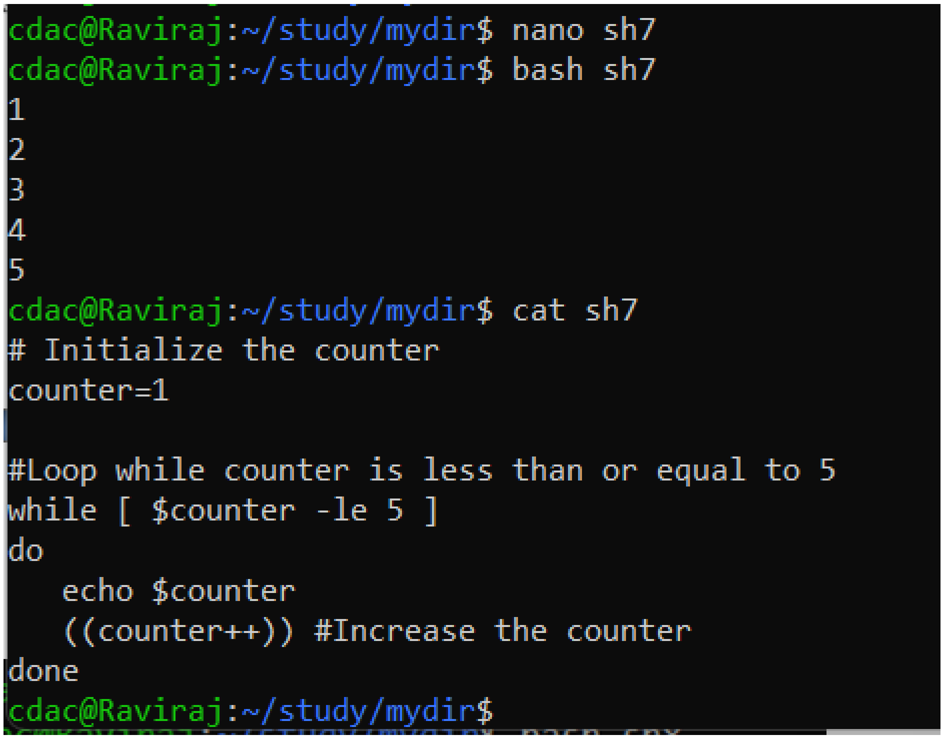
```
cdac@Raviraj:~/study/mydir$ nano sh5
cdac@Raviraj:~/study/mydir$ bash sh5
Enter a number:5
Odd
cdac@Raviraj:~/study/mydir$ bash sh5
Enter a number:8
Even
cdac@Raviraj:~/study/mydir$ cat sh5
#Read a number from user
read -p "Enter a number:" num

# check if the number is even or odd
if ((num % 2 == 0)); then
   echo "Even"
else
   echo "Odd"
fi

cdac@Raviraj:~/study/mydir$
```

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



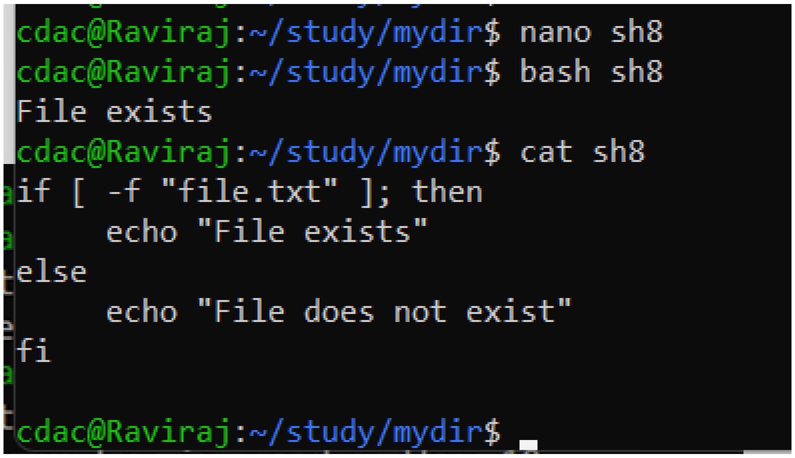
```
cdac@Raviraj:~/study/mydir$ nano sh6
cdac@Raviraj:~/study/mydir$ bash sh6
1
2
3
4
5
cdac@Raviraj:~/study/mydir$ cat sh6
#Loop through numbers 1 to 5
for counter in {1..5}
do
   echo $counter
done
cdac@Raviraj:~/study/mydir$
```

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



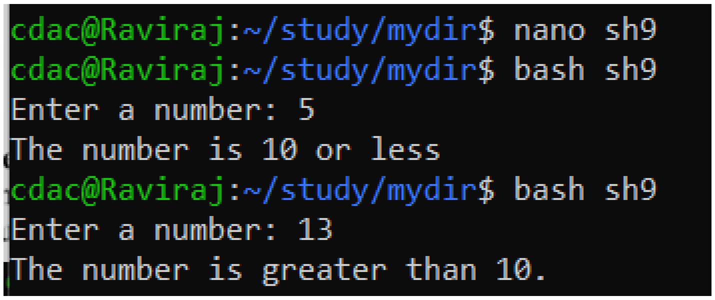
```
cdac@Raviraj:~/study/mydir$ nano sh7
cdac@Raviraj:~/study/mydir$ bash sh7
1
2
3
4
5
cdac@Raviraj:~/study/mydir$ cat sh7
# Initialize the counter
counter=1

#Loop while counter is less than or equal to 5
while [ $counter -le 5 ]
do
   echo $counter
   ((counter++)) #Increase the counter
done
cdac@Raviraj:~/study/mydir$
```

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".



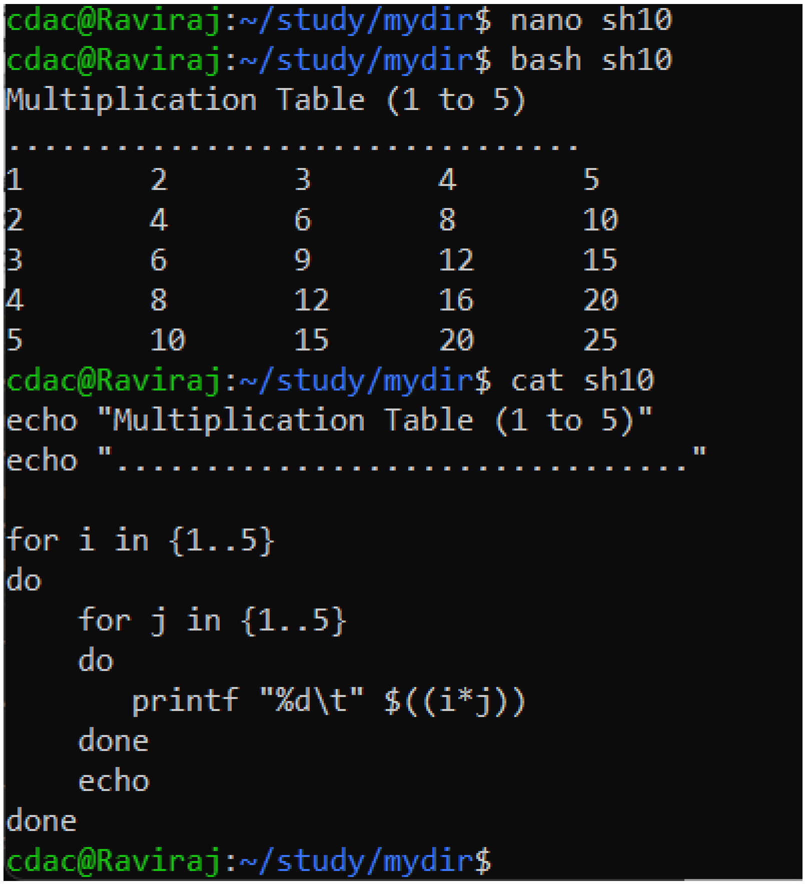
```
cdac@Raviraj:~/study/mydir$ nano sh8
cdac@Raviraj:~/study/mydir$ bash sh8
File exists
cdac@Raviraj:~/study/mydir$ cat sh8
if [ -f "file.txt" ]; then
     echo "File exists"
else
     echo "File does not exist"
fi

cdac@Raviraj:~/study/mydir$ 
```

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.



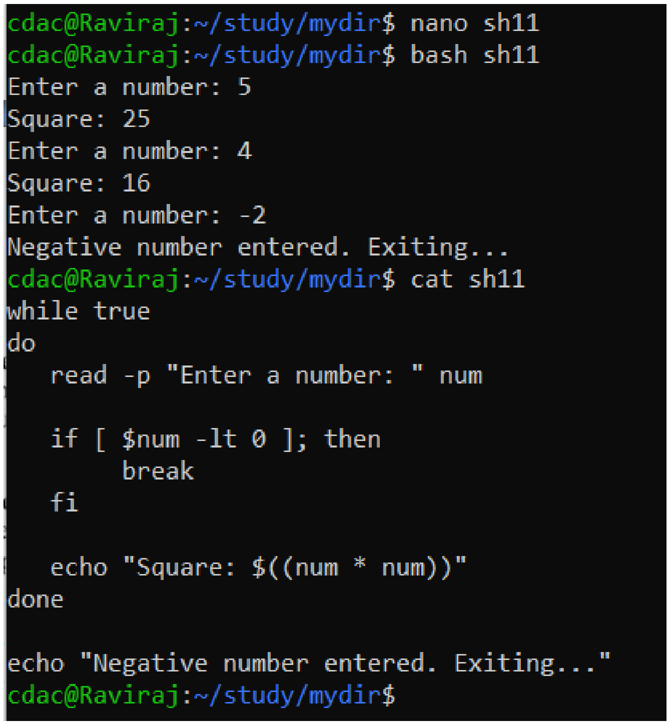
```
cdac@Raviraj:~/study/mydir$ nano sh9
cdac@Raviraj:~/study/mydir$ bash sh9
Enter a number: 5
The number is 10 or less
cdac@Raviraj:~/study/mydir$ bash sh9
Enter a number: 13
The number is greater than 10.
```

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.

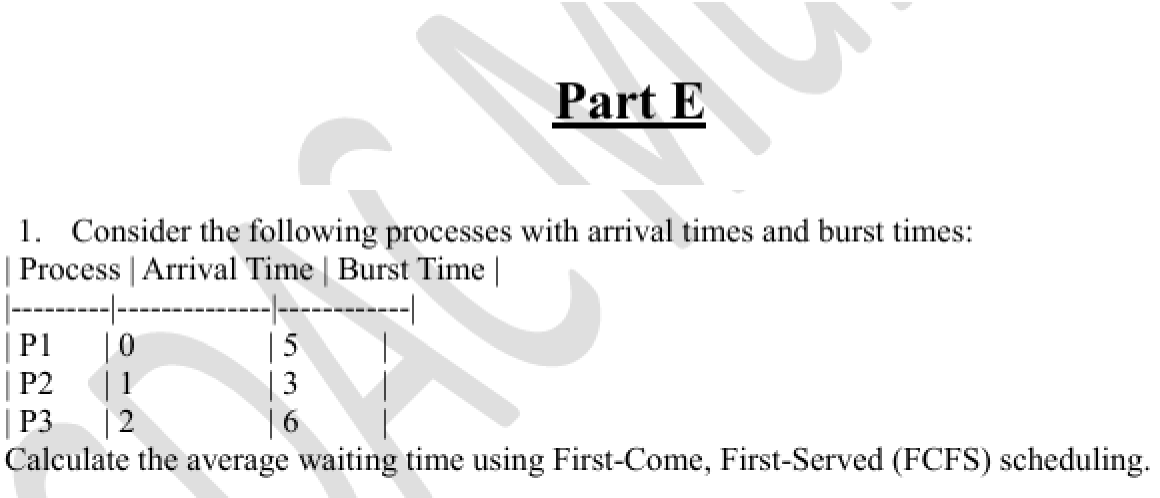


```
cdac@Raviraj:~/study/mydir$ nano sh10
cdac@Raviraj:~/study/mydir$ bash sh10
Multiplication Table (1 to 5)
................................
1       2       3       4       5
2       4       6       8       10
3       6       9       12      15
4       8       12      16      20
5       10      15      20      25
cdac@Raviraj:~/study/mydir$ cat sh10
echo "Multiplication Table (1 to 5)"
echo "................................"

for i in {1..5}
do
    for j in {1..5}
    do
       printf "%d\t" $((i*j))
    done
    echo
done
cdac@Raviraj:~/study/mydir$
```

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@Raviraj:~/study/mydir$ nano sh11
cdac@Raviraj:~/study/mydir$ bash sh11
Enter a number: 5
Square: 25
Enter a number: 4
Square: 16
Enter a number: -2
Negative number entered. Exiting...
cdac@Raviraj:~/study/mydir$ cat sh11
while true
do
   read -p "Enter a number: " num

   if [ $num -lt 0 ]; then
        break
   fi

   echo "Square: $((num * num))"
done

echo "Negative number entered. Exiting..."
cdac@Raviraj:~/study/mydir$
```



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.

Given data:

Process	Arrival Time (AT)	Burst Time	Waiting Time(WT)	TAT
P1	0	5	0	5
P2	1	3	4	7
P3	2	6	6	12

Gantt Chart

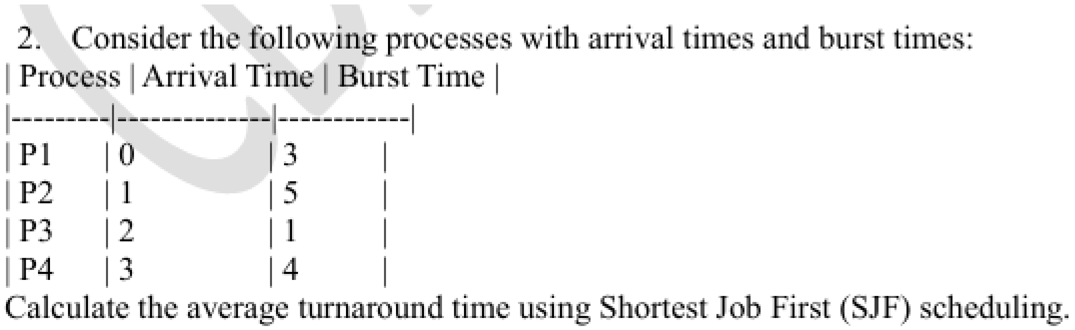
P1	P2	P3	
0	5	8	14

Total waiting time= 0+4+6 =10

Avgerage waiting time= (0+4+6) /3 =10/3 =3.33

turn Around time= 5+7+12=24

Average TurnAround time=(5+7+12)/3 =24/3= 8



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.

Given data:

Process	Arrival Time (AT)	Burst Time	Waiting Time(WT)	TAT
P1	0	3	0+1=1	4
P2	1	5	13	1
P3	2	1	2	3
P4	3	4	4	8

Gantt Chart

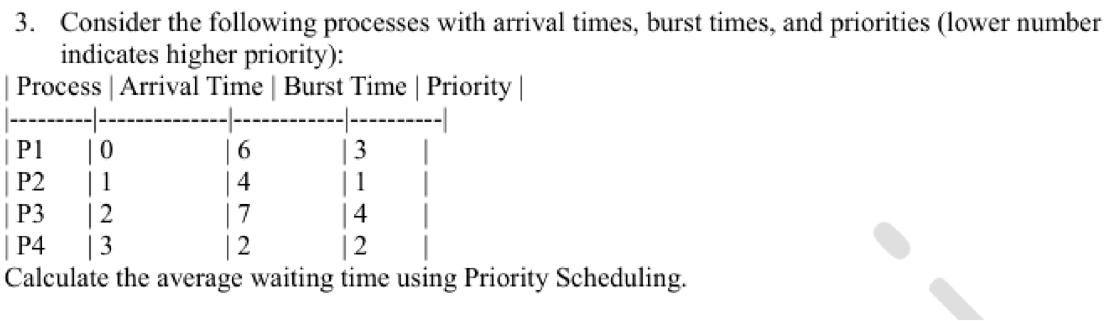
P1	P1	P3	P1	P4	P4	P2
0	1	2	3	4	8	13

Total waiting time= 3+4+6 =10

Avgerage waiting time= (0+4+6) /3 =10/3 =3.33

turn Around time= 5+7+12=24

Average TurnAround time=(5+7+12)/3 =24/3= 8



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.

Given data:

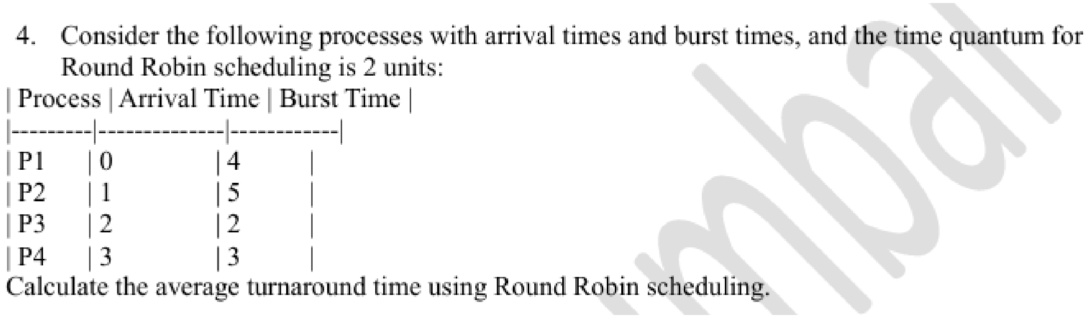
Process	Arrival Time (AT)	Burst Time	Waiting Time(WT)	TAT
P1	0	6	0+7=7	13
P2	1	4	0	4
P3	2	7	11	18
P4	3	2	2	4

Gantt Chart

P1	P2	P4	P1	P3	
0	1	5	7	13	20

Total waiting time= 7+0+11+2 =20

Avgerage waiting time= (7+0+11+2) /4 =20/4 =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.

Given data:

Process	Arrival Time (AT)	Burst Time	Waiting Time(WT)	TAT
P1	0	4	0+6=6	10
P2	1	5	1+5+2 =8	13
P3	2	2	2	4
P4	3	3	3+4=6	10

10

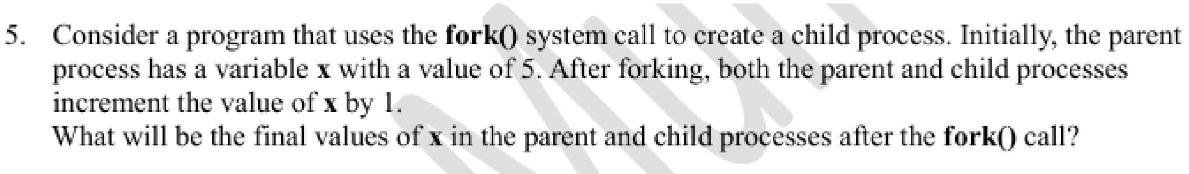
Gantt Chart

P1	P2	P3	P4	P1	P2	P4	P2	
0	2	4	6	8	10	12	13	14

Total Turn Around time= 10+13+4+10 =37

Avgerage waiting time= (10+13+4+10) /4 =37/4 =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?

Given-

Parent x=5
child c= x +1= 5+1=6

After increment by 1 in both parent and child process=
Parent x=5+1=6
child c= 6+1=7

-