

Part A

What will the following commands do?

- `echo "Hello, World!"`
 - ⇒ `echo` is used to send text, variables, and special characters to the standard output.
 - print output "Hello World."

```
cdac@HP:~$ echo "Hello, World"
Hello, World
cdac@HP:~$ |
```

- `name="Productive"`
 - ⇒ `name` is a variable.
 - ⇒ `=` is an assignment operator.
 - ⇒ `Productive` is a string value.
 - ⇒

```
cdac@HP:~/information/assignment$ name=Productivity
cdac@HP:~/information/assignment$ echo $name
Productivity
```

- `touch file.txt`
 - ⇒ `touch` is used to create new file.
 - like as "file.txt".

```
cdac@HP:~/information/assignment$ touch file.txt
cdac@HP:~/information/assignment$ ls
file.txt
cdac@HP:~/information/assignment$ |
```

- `ls -a`
 - ⇒ `ls` is used to list the contents of a directory.
 - ⇒ `ls -a` is also listing the hidden files.
 - ⇒

```
cdac@HP:~/information/assignment$ ls -a
.  ..  file.txt
```

- `rm file.txt`
 - ⇒ `rm` command remove the file.

```
cdac@HP:~/information/assignment$ rm file.txt
cdac@HP:~/information/assignment$ ls
file1.txt
```

- `cp file1.txt file2.txt`
 ⇒ `cp` command copy file or directories to same or different location.

```
cdac@HP:~/information/assignment$ cp file.txt file1.txt
cdac@HP:~/information/assignment$ cat file1.txt
hi
hello
everyone
```

- `mv file.txt /path/to/directory/`
 ⇒ `mv` is used to move files or directories from one location to another.
 ⇒ `/path/to/directory/` this is the destination directory where you want to move the file.

```
cdac@HP:~/information$ mv data.txt assignment
cdac@HP:~/information$ cd assignment
cdac@HP:~/information/assignment$ ls
data.txt file.txt file1.txt
```

- `chmod 755 script.sh`
 ⇒ `chmod` command managing files and directory permissions.
 ⇒ `chmod 755` give the owner full permissions.
 ⇒ `script.sh` is the name of file.

```
cdac@HP:~/information/assignment$ chmod 755 script.sh
cdac@HP:~/information/assignment$ ls -l
total 16
-rw-rwxrwx 1 cdac cdac 114 Aug 29 20:53 data.txt
-rw-r--r-- 1 cdac cdac 19 Aug 30 20:14 file.txt
-rw-r--r-- 1 cdac cdac 19 Aug 30 20:14 file1.txt
-rwxr-xr-x 1 cdac cdac 48 Aug 30 20:36 script.sh
```

- `grep "pattern" file.txt`
 ⇒ `grep` -Searches for a pattern in a file.

```
cdac@HP:~/information$ cat file.txt
hi
pattern
hello
bye
cdac@HP:~/information$ grep "pattern" file.txt
pattern
```

- kill PID
 - ⇒ kill command Sends a signal to a process to terminate it.
- mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt
 - ⇒ mkdir create new directory
 - ⇒ && - It ensures that the next command only runs if the previous command was successful
 - ⇒ cd -to change directory
 - ⇒ touch create new file
 - ⇒ echo Prints a message to the terminal.
 - ⇒ > redirection symbol
 - ⇒ Cat command display content of the file on console.

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

- ls -l | grep ".txt"
 - ⇒ ls -l lists files and directories in the current directory
 - ⇒ ' | ' (pipe) - to redirect the standard output of one command to the standard input of another command.
 - ⇒ grep searches for a patterns in a file.

```
cdac@HP:~/information$ ls -l | grep ".txt"
-rw-r--r-- 1 cdac cdac 28 Aug 29 21:43 duplicate.txt
-rw-r--r-- 1 cdac cdac 22 Aug 30 21:00 file.txt
-rw-r--r-- 1 cdac cdac 60 Aug 29 21:48 fruit.txt
-rw-r--r-- 1 cdac cdac 41 Aug 29 21:24 input.txt
-rw-r--r-- 1 cdac cdac 42 Aug 29 21:04 number.txt
-rw-r--r-- 1 cdac cdac 41 Aug 29 21:30 output.txt
cdac@HP:~/information$ |
```

- cat file1.txt file2.txt | sort | uniq
 - ⇒ cat file1.txt file2.txt concatenate the contents of two files,
 - ⇒ sort the combined contents
 - ⇒ Using uniq to remove duplicate lines.

```

cdac@HP:~/information/mydir$ nano file.txt
cdac@HP:~/information/mydir$ cat file.txt
Hello, World!
hii
good
bye
cdac@HP:~/information/mydir$ nano file1.txt
cdac@HP:~/information/mydir$ cat file1.txt
hi
bye
good
cdac@HP:~/information/mydir$ cat file.txt file1.txt|sort|uniq
Hello, World!
bye
good
hi
hii

```

- `ls -l | grep "^d"`
 - ⇒ `ls -l` list all files and directories
 - ⇒ `grep "^d"` it means matching only lines that start with "d".

```

cdac@HP:~/information/assignment$ ls
dac data.txt day.txt def.txt file.txt file1.txt script.sh
cdac@HP:~/information/assignment$ ls -l | grep "^d"
drwxr-xr-x 2 cdac cdac 4096 Aug 30 21:33 dac

```

- `grep -r "pattern" /path/to/directory/`
 - ⇒ `grep -r "pattern"` it search all files within this directory and its subdirectories for the specified pattern, recursively

```

cdac@HP:~/information/assignment/dac$ grep -r "hi" ~/information/assignment/dac/
/home/cdac/information/assignment/dac/abc.txt:hi
cdac@HP:~/information/assignment/dac$ |

```

- `cat file1.txt file2.txt | sort | uniq -d`
 - ⇒ concatenate both files and display duplicate words only using this command

```
cdac@HP:~/information/assignment$ cat file1.txt
apple
banana
mango
cherry
cdac@HP:~/information/assignment$ cat file.txt
Apple
mango
pinapple
chikku
cdac@HP:~/information/assignment$ cat file.txt file1.txt | sort | uniq -d
mango
```

- `chmod 644 file.txt`
 ⇒ owner can read and write the file or directory and other users can only read it.

```
cdac@HP:~/information/assignment/dac$ chmod 644 file.txt
cdac@HP:~/information/assignment/dac$ ls -l
total 4
-rw-r--r-- 1 cdac cdac 7 Aug 30 21:47 abc.txt
-rw-r--r-- 1 cdac cdac 0 Aug 30 21:54 file.txt
```

- `cp -r source_directory destination_directory`
 ⇒ copy content of source directory to destination directory.

```
cdac@HP:~/information/assignment$ cp -r dac dacs
cdac@HP:~/information/assignment$ ls -r dacs
dac
```

- `find /path/to/search -name "*.txt"`
 ⇒ `find` is used to search for files and directories in directory hierarchy
 ⇒ `/path/to/search` - search actual path to the directory.
 ⇒ `"*.txt"` matches sequence of characters

- `chmod u+x file.txt`
 ⇒ To give permission to owner to execute a file

```
cdac@HP:~/information/assignment/dac$ chmod u+x file.txt
cdac@HP:~/information/assignment/dac$ ls -l
total 4
-rw-r--r-- 1 cdac cdac 7 Aug 30 21:47 abc.txt
-rwxr--r-- 1 cdac cdac 0 Aug 30 21:54 file.txt
cdac@HP:~/information/assignment/dac$ |
```

- `$PATH`
 ⇒ to find the executable file corresponding to that command.

Part B

Identify True or False:

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

Ans: **True** ls command is used list files and directories in directory

2. mv is used to move files and directories.

Ans: **True** mv command used to move files

3. cd is used to copy files and directories.

Ans: **False** cd command used to change directory. Move present directory to another directory.

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** grep command used to search patterns in file.

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** command is used to change file permissions **chmod +x** is for add executable permission

2. cpy is used to copy files and directories.

Ans. No **cpy** command not used to copy files. **cp** command used to copy files

3. mkfile is used to create a new file.

Ans: **mkfile** is used to create new file but it not support for all operating system

4. catx is used to concatenate files.

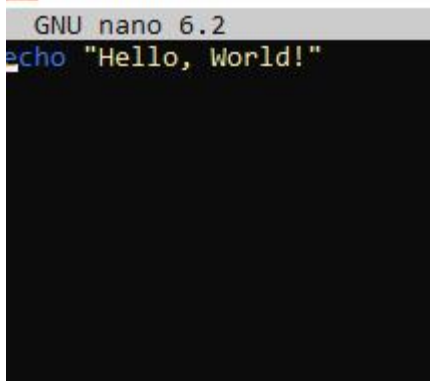
Ans: **cat** command used to concatenate files and also diplay file contents.

5. rn is used to rename files.

Ans **mv** command used to rename files.

Part C

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

A screenshot of a terminal window with a black background. At the top, a grey header bar displays "GNU nano 6.2". Below the header, the text "echo "Hello, World!"" is written in a light blue monospace font. A white cursor is positioned at the end of the line.

```
GNU nano 6.2
echo "Hello, World!"
```


Output:

```
cdac@DESKTOP-5IN0EGJ:~$ nano p1.sh
cdac@DESKTOP-5IN0EGJ:~$ bash p1.sh
Hello, World!
cdac@DESKTOP-5IN0EGJ:~$ _
```

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

```
cdac@DESKTOP-5IN0EGJ: ~
GNU nano 6.2
Name="CDAC Mumbai"
echo $Name
_
```

Output:

```
cdac@DESKTOP-5IN0EGJ:~$ nano p2.sh
cdac@DESKTOP-5IN0EGJ:~$ bash p2.sh
CDAC Mumbai
cdac@DESKTOP-5IN0EGJ:~$
```


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

```
cdac@DESKTOP-5IN0EGJ: ~  
GNU nano 6.2  
echo "Enter Number:"  
read Num1  
echo Number is: $Num1
```

Output:

```
cdac@DESKTOP-5IN0EGJ: ~  
cdac@DESKTOP-5IN0EGJ:~$ nano p3  
cdac@DESKTOP-5IN0EGJ:~$ bash p3  
Enter Number:  
23  
Number is: 23  
cdac@DESKTOP-5IN0EGJ:~$
```

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

```
cdac@DESKTOP-5IN0EGJ: ~  
GNU nano 6.2  
#!/bin/bash  
echo Enter 1st Number:  
read num1  
echo Enter 2nd Number:  
read num2  
sum=$(( $num1 + $num2 ))  
echo Addition of Two Number is: $sum
```

Output

```
cdac@DESKTOP-5IN0EGJ:~$ nano p4
cdac@DESKTOP-5IN0EGJ:~$ bash p4
Enter 1st Number:
23
Enter 2nd Number:
25
Addition of Two Number is: 48
cdac@DESKTOP-5IN0EGJ:~$
```

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

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

echo Enter a number:
read num1
if [  $((\$num1 \% 2)) -eq 0$  ]
then
echo $num1 is Even Number
else
echo $num1 is odd Number
fi
```

Output

```
cdac@DESKTOP-5IN0EGJ:~$ nano p5
cdac@DESKTOP-5IN0EGJ:~$ bash p5
Enter a number:
23
23 is odd Number
cdac@DESKTOP-5IN0EGJ:~$ bash p5
Enter a number:
44
44 is Even Number
cdac@DESKTOP-5IN0EGJ:~$
```

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

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

echo Number from 1 to 5
for (( i=1; i<=5 ; i++))
do
echo $i
done
```

Output:

```
cdac@DESKTOP-5IN0EGJ:~$ nano p6
cdac@DESKTOP-5IN0EGJ:~$ bash p6
Number from 1 to 5
1
2
3
4
5
cdac@DESKTOP-5IN0EGJ:~$
```

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

```
cdac@DESKTOP-5IN0EGJ: ~
GNU nano 6.2
#!/bin/bash

echo Numbers from 1 to 5
i=1

while (( i<=5 ))
do
echo $i
i=$((i+1))
done
```

Output:

```
cdac@DESKTOP-5IN0EGJ: ~  
cdac@DESKTOP-5IN0EGJ:~$ nano p7  
cdac@DESKTOP-5IN0EGJ:~$ bash p7  
Numbers from 1 to 5  
1  
2  
3  
4  
5  
cdac@DESKTOP-5IN0EGJ:~$
```

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".

```
GNU nano 6.2  
#!/bin/bash  
  
filename="/home/cdac/abc.txt"  
if [ -f $filename ]  
then  
echo "File exists"  
else  
echo "File does not exists"  
fi  
_
```

Output:

```
cdac@DESKTOP-5IN0EGJ: ~  
cdac@DESKTOP-5IN0EGJ:~$ nano p8  
cdac@DESKTOP-5IN0EGJ:~$ bash p8  
File exists  
cdac@DESKTOP-5IN0EGJ:~$ nano p8  
cdac@DESKTOP-5IN0EGJ:~$ bash p8  
File does not exists  
cdac@DESKTOP-5IN0EGJ:~$ _
```

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
#!/bin/bash
echo Enter a Number:
read num1
if [ $num1 -gt 10 ]
then
echo Number is greater than 10
else
echo Number not grater than 10
fi
_
```

Output:

```
cdac@DESKTOP-5IN0EGJ: ~
cdac@DESKTOP-5IN0EGJ:~$ nano p9
cdac@DESKTOP-5IN0EGJ:~$ bash p9
Enter a Number:
5
Number not grater than 10
cdac@DESKTOP-5IN0EGJ:~$ bash p9
Enter a Number:
23
Number is greater than 10
cdac@DESKTOP-5IN0EGJ:~$ _
```

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
echo Multiplication Table for Numbers from 1 to 5:
for (( i=1;i<=10;i++ ))
do
for (( j=1;j<=10;j++ ))
do
result=$((i*j))
printf "%4d" $result
done
echo
done
```

Output:

```
cdac@DESKTOP-5IN0EGJ:~$ nano p10
cdac@DESKTOP-5IN0EGJ:~$ bash p10
Multiplication Table for Numbers from 1 to 10:
 1  2  3  4  5  6  7  8  9 10
 2  4  6  8 10 12 14 16 18 20
 3  6  9 12 15 18 21 24 27 30
 4  8 12 16 20 24 28 32 36 40
 5 10 15 20 25 30 35 40 45 50
 6 12 18 24 30 36 42 48 54 60
 7 14 21 28 35 42 49 56 63 70
 8 16 24 32 40 48 56 64 72 80
 9 18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100
cdac@DESKTOP-5IN0EGJ:~$
```

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.

```
GNU nano 6.2
#!/bin/bash
while true
do
echo Enter a Number
read num1
if [ $num1 -lt 0 ]
then
    echo Negative Number Entered. Exiting.
    break
fi
square=$((num1*num1))
echo The Squareof $num1 is $square
done
_
```

Output

```
cdac@DESKTOP-5IN0EGJ: ~
cdac@DESKTOP-5IN0EGJ:~$ nano p11
cdac@DESKTOP-5IN0EGJ:~$ bash p11
Enter a Number
4
The Squareof 4 is 16
Enter a Number
3
The Squareof 3 is 9
Enter a Number
11
The Squareof 11 is 121
Enter a Number
-44
Negative Number Entered. Exiting.
cdac@DESKTOP-5IN0EGJ:~$ _
```


Part E

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

| Process | Arrival Time | Burst Time |

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

| P1 | 0 | 5 |

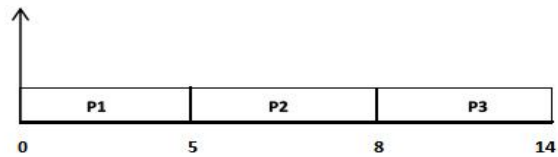
| P2 | 1 | 3 |

| P3 | 2 | 6 |

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

	Arrival Time	Burst Time	Waiting Time	Turn Around Time
P1	0	5	0	5
P2	1	3	4	7
P3	2	6	6	12

:



Avg Waiting Time- $10/3=3.3$

TAT= $5+7+12=24/3=8$

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 |

Shortest Job First				
	AT	BT	W	TAT
P1	0	3	0	3
P2	1	5	7	12
P3	2	1	1	2
P4	3	4	1	5

Gant Chart				
P1	P3	P4	P2	
0	3	4	8	13
TAT		3+12+2+5	22 /4	5.5

indicates higher priority):

A horizontal number line with four equal segments. Each segment contains a dashed line for a digit. The segments are separated by vertical tick marks.

| P2 | 1 | 4 | 1 |

| P4 | 3 | 2 | 2 |

Calculate the average waiting time using Priority Scheduling.

Priority Scheduling Algorithm					
	AT	BT	PRIORITY	WAT	TAT
P1	0	6	3	3	12
P2	1	4	1	12	4
P3	2	7	4	2	17
P4	3	2	2	5	4

Gant Chart		P1	P2	P4	P1	P3
	0	1	5	7	12	19
TAT	12+4+17+4	37 /4	9.2			

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	10
P2	1	4
P3	2	6
P4	4	2
P5	5	4

| P1 | 0 | 4 |

| P2 | 1 | 5 |

P3	2	2
----	---	---

| P4 | 3 | 3 |

Calculate the average turnaround time using Round Robin scheduling.

Round Robin Algorithm				
	AT	BT	WAT	TAT
P1	0	4	3	10
P2	1	5	12	13
P3	2	2	2	4
P4	3	3	5	10

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

TAT	
12+4+17+4	37 / 4
	9.2

