PART - A

What will the following commands do?

1. **echo "Hello, World!":**

Prints the string "Hello, World!" to the terminal.

```
cdac@LAPTOP-184M49M6:~$ echo "Hello, World!"
Hello, World!
cdac@LAPTOP-184M49M6:~$
```

2. **name="Productive":**

Assigns the value "Productive" to the variable `name`.

3. **touch file.txt:**

Creates a new, empty file named `file.txt` if it doesn't already exist, or updates the timestamp of `file.txt` if it does exist.

4. **ls -a:**

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

```
cdac@LAPTOP-184M49M6:~$ touch file.txt
cdac@LAPTOP-184M49M6:~$ ls -a
.               .cache        .sudo_as_admin_successful  file.txt
..              .config       Day1                       p7
.bash_history   .local        ShellProgramming           wildcard
.bash_logout    .motd_shown   abc.txt
.bashrc         .profile      dir4
cdac@LAPTOP-184M49M6:~$
```

5. **rm file.txt:**

Deletes the file named `file.txt`.

```
cdac@LAPTOP-184M49M6:~$ rm file.txt
cdac@LAPTOP-184M49M6:~$ ls
Day1  ShellProgramming  abc.txt  dir4  p7  wildcard
cdac@LAPTOP-184M49M6:~$
```

6. **cp file1.txt file2.txt:**

Copies the contents of `file1.txt` to `file2.txt`. If `file2.txt` doesn't exist, it will be created.

```
cdac@LAPTOP-184M49M6:~$ nano file1.txt
cdac@LAPTOP-184M49M6:~$ cat file1.txt
Priyanka Bolaj
cdac@LAPTOP-184M49M6:~$ cp file1.txt file2.txt
cdac@LAPTOP-184M49M6:~$ cat file2.txt
Priyanka Bolaj
cdac@LAPTOP-184M49M6:~$
```

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

Moves `file.txt` to the specified directory (`/path/to/directory/`). It can also rename the file if a new filename is given in the destination path.

```
cdac@LAPTOP-184M49M6:~$ mv file1.txt Day1
cdac@LAPTOP-184M49M6:~$ ls Day1
file1.txt
cdac@LAPTOP-184M49M6:~$
```

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

Changes the permissions of the file `script.sh` to `755`, which makes it readable and executable by everyone, but writable only by the owner.

```
cdac@LAPTOP-184M49M6:~$ chmod 755 file2.txt
cdac@LAPTOP-184M49M6:~$ ls -l
total 28
drwxr-xr-x 2 cdac cdac 4096 Sep  1 10:55 Day1
drwxr-xr-x 2 cdac cdac 4096 Aug 31 12:23 ShellProgramming
-rw-rw-r-- 1 cdac cdac    7 Aug 27 19:11 abc.txt
drwxr-xr-x 3 cdac cdac 4096 Aug 27 20:27 dir4
-rwxr-xr-x 1 cdac cdac   15 Sep  1 10:52 file2.txt
-rw-r--r-- 1 cdac cdac  175 Aug 31 18:42 p7
drwxr-xr-x 2 cdac cdac 4096 Aug 31 11:00 wildcard
cdac@LAPTOP-184M49M6:~$
```

9. **grep "pattern" file.txt:**

Searches for the string "pattern" in `file.txt` and prints the lines that contain it.

```
cdac@LAPTOP-184M49M6:~$ grep "Bolaj" file2.txt
Priyanka Bolaj
cdac@LAPTOP-184M49M6:~$
```

10. kill PID:

Terminates the process with the specified Process ID (`PID`).

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

This sequence of commands:

- Creates a new directory called `mydir`.

- Changes the current directory to `mydir`.

- Creates a new file called `file.txt`.

- Writes "Hello, World!" into `file.txt`.

- Displays the contents of `file.txt`.

```
cdac@LAPTOP-184M49M6:~$ mkdir mydir && cd mydir && touch file.tx
t && echo "Hello, World!" > file.txt && cat file.txt
Hello, World!
cdac@LAPTOP-184M49M6:~/mydir$
```

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

Lists all files in the current directory with detailed information (`ls -l`), and then filters the results to show only the lines containing `.txt`, which typically represent text files.

```
cdac@LAPTOP-184M49M6:~/mydir$ ls -l | grep ".txt"
-rw-r--r-- 1 cdac cdac 14 Sep  1 10:59 file.txt
cdac@LAPTOP-184M49M6:~/mydir$
```

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

Combines the contents of `file1.txt` and `file2.txt`, sorts them, and removes duplicate lines, displaying only unique lines.

```
cdac@LAPTOP-184M49M6:~$ nano file1.txt
cdac@LAPTOP-184M49M6:~$ nano file2.txt
cdac@LAPTOP-184M49M6:~$ cat file1.txt file2.txt | sort | uniq
Bolaj
Milind
Neha
Priya
Priyanka Bolaj
cvbher
sdvsd
sdvsdf
tyerfn
cdac@LAPTOP-184M49M6:~$
```

14. **ls -l | grep "^d":**

Lists all files and directories in the current directory with detailed information (`ls -l`), and then filters the results to show only directories (lines starting with `d`).

```
cdac@LAPTOP-184M49M6:~$ ls -l | grep "^d"
drwxr-xr-x 2 cdac cdac 4096 Sep  1 10:55 Day1
drwxr-xr-x 2 cdac cdac 4096 Aug 31 12:23 ShellProgramming
drwxr-xr-x 3 cdac cdac 4096 Aug 27 20:27 dir4
drwxr-xr-x 2 cdac cdac 4096 Sep  1 10:59 mydir
drwxr-xr-x 2 cdac cdac 4096 Aug 31 11:00 wildcard
```

la

15. **grep -r "pattern" /path/to/directory/:**

Recursively searches for the string "pattern" in all files within `/path/to/directory/` and its subdirectories.

16. **cat file1.txt file2.txt | sort | uniq -d:**

Combines the contents of `file1.txt` and `file2.txt`, sorts them, and displays only duplicate lines (lines that appear more than once).

17. **chmod 644 file.txt:**

Changes the permissions of `file.txt` to `644`, making it readable by everyone, but writable only by the owner.

18. **cp -r source_directory destination_directory:**

Recursively copies the `source_directory` and all of its contents to `destination_directory`.

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

Searches for all files ending with `.txt` within `/path/to/search` and its subdirectories.

20. chmod u+x file.txt:

Adds execute permission for the owner of `file.txt`.

```
cdac@LAPTOP-184M49M6:~$ ls
Day1  ShellProgramming  abc.txt  dir4  file1.txt  file2.txt  mydir  p7  wildcard
cdac@LAPTOP-184M49M6:~$ chmod u+x abc.txt
cdac@LAPTOP-184M49M6:~$ ls -l
total 36
drwxr-xr-x 2 cdac cdac 4096 Sep  1 10:55 Day1
drwxr-xr-x 2 cdac cdac 4096 Aug 31 12:23 ShellProgramming
-rwxrw-r-- 1 cdac cdac    7 Aug 27 19:11 abc.txt
drwxr-xr-x 3 cdac cdac 4096 Aug 27 20:27 dir4
-rw-r--r-- 1 cdac cdac   27 Sep  1 11:01 file1.txt
-rwxr-xr-x 1 cdac cdac   39 Sep  1 11:02 file2.txt
drwxr-xr-x 2 cdac cdac 4096 Sep  1 10:59 mydir
-rw-r--r-- 1 cdac cdac  175 Aug 31 18:42 p7
drwxr-xr-x 2 cdac cdac 4096 Aug 31 11:00 wildcard
cdac@LAPTOP-184M49M6:~$
```

21. echo $PATH:

Displays the current value of the `PATH` environment variable, which contains directories where executable files are located.

```
cdac@LAPTOP-184M49M6:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/lib/wsl/lib:/mnt/c/oracle
e/app/oracle/product/11.2.0/server/bin:/mnt/c/Program Files/Common Files/Oracle/Java/javapath:/mnt/c/WINDOWS/system32:/m
nt/c/WINDOWS:/mnt/c/WINDOWS/System32/Wbem:/mnt/c/WINDOWS/System32/WindowsPowerShell/v1.0/:/mnt/c/WINDOWS/System32/OpenSS
H/:/mnt/c/Program Files (x86)/Microsoft SQL Server/160/Tools/Binn/:/mnt/c/Program Files/Microsoft SQL Server/160/Tools/B
inn/:/mnt/c/Program Files/Microsoft SQL Server/Client SDK/ODBC/170/Tools/Binn/:/mnt/c/Program Files/Microsoft SQL Server
/160/DTS/Binn/:/mnt/c/Program Files (x86)/Microsoft SQL Server/160/DTS/Binn/:/mnt/c/Program Files (x86)/Microsoft SQL Se
rver/110/Tools/Binn/ManagementStudio/:/mnt/c/Program Files (x86)/Microsoft SQL Server/110/Tools/Binn/:/mnt/c/Program Fil
es (x86)/Microsoft Visual Studio 11.0/Common7/IDE/PrivateAssemblies/:/mnt/c/Program Files (x86)/Microsoft SQL Server/110
/DTS/Binn/:/mnt/c/Program Files/Microsoft SQL Server/130/Tools/Binn/:/mnt/c/Program Files (x86)/Microsoft SQL Server/120
/DTS/Binn/:/mnt/c/Program Files (x86)/Microsoft SQL Server/130/DTS/Binn/:/mnt/c/Program Files (x86)/Microsoft SQL Server
/140/DTS/Binn/:/mnt/c/Program Files (x86)/Microsoft SQL Server/150/DTS/Binn/:/mnt/c/Program Files/dotnet/:/mnt/c/Program
 Files/Git/cmd:/mnt/c/Program Files/Java/jdk-18.0.2/bin:/mnt/c/Users/Priyanka/AppData/Local/Programs/Microsoft VS Code/b
in:/mnt/c/Program Files/Azure Data Studio/bin:/mnt/c/Program Files/JetBrains/IntelliJ IDEA 2023.3.2/bin:/mnt/c/Users/Pri
yanka/AppData/Local/Microsoft/WindowsApps:/snap/bin
cdac@LAPTOP-184M49M6:~$
```

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: `cd` is used to change the current directory, not to copy files and directories. The `cp` command is used for copying.

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.
   Incorrect
   Ans: - chmod is used to change file permissions.

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

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

4. catx is used to concatenate files.
   Incorrect
   Ans: - cat is used to concatenate files.

5. rn is used to rename files.
   Incorrect
   Ans: - mv is used to rename files.

PART – C

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

```
  GNU nano 6.2                          p1
#!/bin/bash
echo "Hello, World!"
```

```
cdac@LAPTOP-184M49M6:~/OSAssignment2$ nano p1
cdac@LAPTOP-184M49M6:~/OSAssignment2$ bash p1
Hello, World!
cdac@LAPTOP-184M49M6:~/OSAssignment2$
```

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

```
  GNU nano 6.2                          p2
#!/bin/bash
name="CDAC Mumbai"
echo $name
```

```
cdac@LAPTOP-184M49M6:~/OSAssignment2$ nano p2
cdac@LAPTOP-184M49M6:~/OSAssignment2$ bash p2
CDAC Mumbai
cdac@LAPTOP-184M49M6:~/OSAssignment2$
```

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

```
  GNU nano 6.2                          p3
#!/bin/bash
echo "Enter a number:"
read number
echo "You entered: $number"
```

```
cdac@LAPTOP-184M49M6:~/OSAssignment2$ nano p3
cdac@LAPTOP-184M49M6:~/OSAssignment2$ bash p3
Enter a number:
5
You entered: 5
cdac@LAPTOP-184M49M6:~/OSAssignment2$
```

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

```
  GNU nano 6.2                          p4
#!/bin/bash
num1=5
num2=3
sum=$((num1 + num2))
echo "The sum is: $sum"
```

```
cdac@LAPTOP-184M49M6:~/OSAssignment2$ nano p4
cdac@LAPTOP-184M49M6:~/OSAssignment2$ bash p4
The sum is: 8
cdac@LAPTOP-184M49M6:~/OSAssignment2$
```

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                          p5
#!/bin/bash
echo "Enter a number:"
read number
if (( number % 2 == 0 )); then
    echo "Even"
else
    echo "Odd"
fi
```

```
cdac@LAPTOP-184M49M6:~/OSAssignment2$ nano p5
cdac@LAPTOP-184M49M6:~/OSAssignment2$ bash p5
Enter a number:
5
Odd
cdac@LAPTOP-184M49M6:~/OSAssignment2$ bash p5
Enter a number:
4
Even
cdac@LAPTOP-184M49M6:~/OSAssignment2$
```

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

```
  GNU nano 6.2                          p6
#!/bin/bash
for i in {1..5}
do
    echo $i
done
```

```
cdac@LAPTOP-184M49M6:~/OSAssignment2$ nano p6
cdac@LAPTOP-184M49M6:~/OSAssignment2$ bash p6
1
2
3
4
5
cdac@LAPTOP-184M49M6:~/OSAssignment2$
```

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

```
  GNU nano 6.2                          p7 *
#!/bin/bash
i=1
while [ $i -le 5 ]
do
    echo $i
    ((i++))
done
```

```
cdac@LAPTOP-184M49M6:~/OSAssignment2$ nano p7
cdac@LAPTOP-184M49M6:~/OSAssignment2$ bash p7
1
2
3
4
5
cdac@LAPTOP-184M49M6:~/OSAssignment2$
```

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                          p8
#!/bin/bash
if [ -f "file.txt" ]; then
    echo "File exists"
else
    echo "File does not exist"
fi
```

```
cdac@LAPTOP-184M49M6:~/OSAssignment2$ nano p8
cdac@LAPTOP-184M49M6:~/OSAssignment2$ bash p8
File does not exist
cdac@LAPTOP-184M49M6:~/OSAssignment2$ touch file.txt
cdac@LAPTOP-184M49M6:~/OSAssignment2$ bash p8
File exists
cdac@LAPTOP-184M49M6:~/OSAssignment2$
```

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                          p9
#!/bin/bash
echo "Enter a number:"
read number
if [ $number -gt 10 ]; then
    echo "The number is greater than 10"
else
    echo "The number is 10 or less"
fi
```

```
cdac@LAPTOP-184M49M6:~/OSAssignment2$ nano p9
cdac@LAPTOP-184M49M6:~/OSAssignment2$ bash p9
Enter a number:
11
The number is greater than 10
cdac@LAPTOP-184M49M6:~/OSAssignment2$ bash p9
Enter a number:
9
The number is 10 or less
cdac@LAPTOP-184M49M6:~/OSAssignment2$
```

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                          p10
#!/bin/bash
for i in {1..10}
do
    for j in {1..5}
    do
        result=$((i * j))
        printf "%4d" $result
    done
    echo
done
```

```
cdac@LAPTOP-184M49M6:~/OSAssignment2$ nano p10
cdac@LAPTOP-184M49M6:~/OSAssignment2$ bash p10
   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
   6  12  18  24  30
   7  14  21  28  35
   8  16  24  32  40
   9  18  27  36  45
  10  20  30  40  50
cdac@LAPTOP-184M49M6:~/OSAssignment2$
```

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                          p11
#!/bin/bash
while :
do
    echo "Enter a number (negative number to exit):"
    read number
    if [ $number -lt 0 ]; then
        break
    fi
    square=$((number * number))
    echo "Square: $square"
done
```

```
cdac@LAPTOP-184M49M6:~/OSAssignment2$ nano p11
cdac@LAPTOP-184M49M6:~/OSAssignment2$ bash p11
Enter a number (negative number to exit):
5
Square: 25
Enter a number (negative number to exit):
7
Square: 49
Enter a number (negative number to exit):
-6
cdac@LAPTOP-184M49M6:~/OSAssignment2$
```

Assignment 2

M T W T F S S
Page No.:
Date:
YOUVA

Part - E

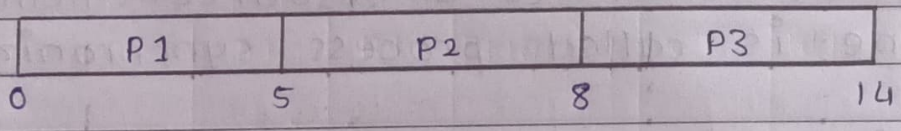
Q1. Consider the following processes with arrival times and burst time.

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.

⟹

Gxantt chart



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

Average waiting time

$$= \frac{0+4+6}{3} = \frac{10}{3} = 3.33$$

Average TAT $= \frac{5+7+12}{3} = 8$

Q2. Consider the following processes with arrival time & 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.

⇒ Gantt Chart

P1	P3	P4	P2
0	3	4	8	13

Process	A.T	B.T.	Completion time	Waiting time	TAT CT-AT
P1	0	3	3	0	3
P2	1	5	13	7	12
P3	2	1	4	1	2
P4	3	4	8	1	5

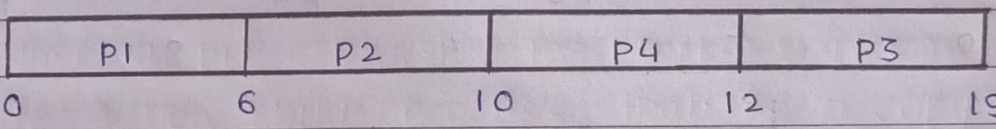
Average TAT = $\frac{3+12+2+5}{4} = \frac{22}{4} = 5.5$

Q3. Consider the following processes with arrival times, burst time 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.

⟹ Grantt chart



Process	AT	BT	Priority	CT	WT	TAT CT-AT
P1	0	6	3	6	0	0
P2	1	4	1	10	5	9
P3	2	7	4	19	10	17
P4	3	2	2	12	7	9

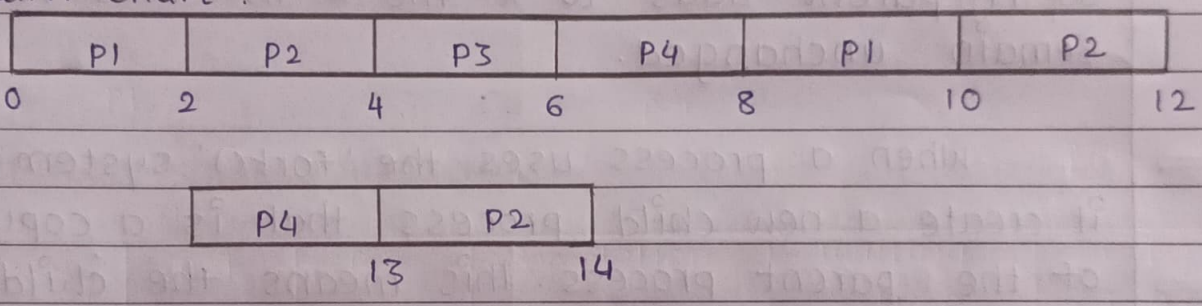
Average WT = $\frac{0+5+10+7}{4} = \frac{22}{4} = 5.5$

Q4 Consider the following processes with arrival times & burst time & 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.

⇒ Gantt Chart :



Process	AT	BT	CT	waiting Time	TAT CT-AT
P1	0	4	10	6	10
P2	1	5	14	(1+6+1) 8	13
P3	2	2	6	2+ 2	4
P4	3	3	13	3+4 7	10

Average TAT = $\frac{10+13+4+10}{4} = \frac{37}{4} = 9.25$

Q5. 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 & child processes increment the value of x by 1.
What will be the final values of x in the parent & child processes after the fork() call?

⟹ x = 6

Child and parent process maintain separate copies of all the variables & can't communicate so if parent adds to x the x in child will remain unchanged.

When a process uses the 'fork()' system call, it create a new child process that is a copy of the parent process. This means the child process gets its own copy of all the variables, including 'x'.
Step by step:
1) The parent process has variable 'x' with a value of '5'.
2) The 'fork()' call creates a child process. At this point, both the parent & child processes have their own separate copies of 'x' & both have 'x=5'.
3) After the fork, both the parent & child processes independently increment their own copy of 'x' by '1'. In the parent process, 'x' becomes '6'. In child process, 'x' becomes '6'.
↳ Process Parent :- 'x=6' ↳ Child process :- 'x=6'
↳ The parent & child processes have separate copies of the variable 'x'. Changes in one process do not affect the other. Therefore, after both process increment 'x', the value of 'x' in both the parent & child processes will be '6'.