Concepts of Operating System
Assignment 2

Part A

What will the following commands do?

- echo "Hello, World!"

It will print Hello World. ‘echo’ commands is used whenever user have anything to Print in this case it is Hello World, it can also take file and print it.

- name="Productive"

This command is used to assign string literal.

- touch file.txt

This command will create empty file. 'touch’ command is used to create new file in this case the file was file.txt, file.txt is file name.

- ls –a

It is used to list the contents of acurrent directory.
With –a we can list hidden files and directories.

- rm file.txt

This command removes (deletes) the file file.txt. Be careful, as this is permanent!

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

This moves (renames) file.txt to the specified directory /path/to/directory/. If a file with the same name already exists in the destination directory, it will be overwritten.

- chmod 755 script.sh

chmod changes the permissions of a file. 755 represents permissions in octal notation.
7 (owner): read, write, execute
5 (group): read, execute
5 (others): read, execute
This command typically makes script.sh executable.

- grep "pattern" file.txt

grep searches for lines in file.txt that contain the specified "pattern" and prints those lines to the standard output

- kill PID

kill sends a signal to the process with the given process ID (PID). By default, it sends the TERM (terminate) signal, which asks the process to gracefully exit.

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

This is a series of commands chained together with && (execute the next command only if the previous one succeeds).
-mkdir mydir: Creates a 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!" to file.txt,

overwriting any existing content.
-cat file.txt: Displays the contents of file.txt (which will be "Hello, World!").

- ls -l | grep ".txt"

ls -l: Lists files and directories in long format.
grep "^d": Filters the output, showing only lines that start with "d" (indicating directories). ^ matches the beginning of a line.

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

cat file1.txt file2.txt: Concatenates the contents of file1.txt and file2.txt and sends the combined output to the pipe.

sort: Sorts the lines of the input.
uniq: Removes duplicate lines from the sorted input.

- ls -l | grep "^d"

ls -l: Lists files and directories in long format.
grep "^d": Filters the output, showing only lines that start with "d" (indicating directories). ^ matches the beginning of a line.

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

grep: Searches for the "pattern".
-r: Recursive search, meaning it searches within all subdirectories of /path/to/directory/.

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

cat file1.txt file2.txt: Concatenates the contents of file1.txt and file2.txt.
sort: Sorts the lines.
uniq -d: Displays only the duplicate lines from the sorted input.

- chmod 644 file.txt

chmod: changes file permissions.
644:
6 (owner): read, write
4 (group): read
4 (others): read
This is a common permission setting for data files.

- cp -r source_directory destination_directory

cp: copies files and directories.
-r: Recursive copy, meaning it copies the source_directory and all its contents (including subdirectories) to destination_directory

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

find: Searches for files and directories.
/path/to/search: The directory to start the search in.
-name "*.txt": Finds files with names that match the pattern "*.txt" (i.e., files ending in ".txt").

- chmod u+x file.txt

chmod: changes file permissions.
u+x: Adds execute permission for the file owner (u for user).
echo $PATH

- echo $PATH

echo: Displays text.
$PATH: Refers to the shell variable PATH, which contains a colon-separated list of directories where the shell looks for executable files. This command displays the current PATH environment variable.

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.- **False**

Identify the Incorrect Commands:

1. chmodx is used to change file permissions. -**Incorrect**
2. cpy is used to copy files and directories. - **Incorrect**
3. mkfile is used to create a new file.- **Incorrect**
4. catx is used to concatenate files.- **Incorrect**
5. rn is used to rename files.- **Incorrect**

Part C

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

```
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$ chmod +x Assin.sh
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$ nano Assin.sh
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$ ./Assin.sh
Hello, World!
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$
```

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

```
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$ nano Assin.sh
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$ ./Assin.sh
CDAC Mumbai
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$
```

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

```
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$ nano Assin.sh
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$ cat Assin.sh

echo "Enter a number"
read a
echo Your number is $a

cdac@DESKTOP-R3VB19M:~/LinuxAssignment$ ./Assin.sh
Enter a number
456
Your number is 456
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$
```

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

```
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$ nano Assin.sh
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$ cat Assin.sh

echo "Enter a number"
read a
echo "Enter a number"
read b
sum='expr $a + $b'
echo sum of $a + $b is $sum

cdac@DESKTOP-R3VB19M:~/LinuxAssignment$ ./Assin.sh
Enter a number
45
Enter a number
23
sum of 45 + 23 is expr $a + $b
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$
```

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

```
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$ nano Assin.sh
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$ cat Assin.sh

echo "Enter a number"
read a
if [ 'expr $a % 2' -eq 0 ]
then
      echo "$a is an even number"
else
      echo "$a is odd number"
fi


cdac@DESKTOP-R3VB19M:~/LinuxAssignment$ ./Assin.sh
Enter a number
24
./Assin.sh: line 4: [: expr $a % 2: integer expression expected
24 is odd number
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$
```

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

```
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$ nano Assin.sh
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$ cat Assin.sh
for i in 1 2 3 4 5
do
    echo $i
done



cdac@DESKTOP-R3VB19M:~/LinuxAssignment$ ./Assin.sh
1
2
3
4
5
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$
```

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

```
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$ nano Assin.sh
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$ cat Assin.sh
a=1
while [ $a -lt 6 ]
do
      echo $a
a='expr $a + 1'
done



cdac@DESKTOP-R3VB19M:~/LinuxAssignment$ ./Assin.sh
1
./Assin.sh: line 2: [: too many arguments
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$
```

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@DESKTOP-R3VB19M:~/LinuxAssignment$ nano Assin.sh
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$ cat Assin.sh
if [ -e file.txt ]
then
       echo "File exists"
else
       echo "File doesn't exist"
fi


cdac@DESKTOP-R3VB19M:~/LinuxAssignment$ ./Assin.sh
File doesn't exist
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$
```

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.

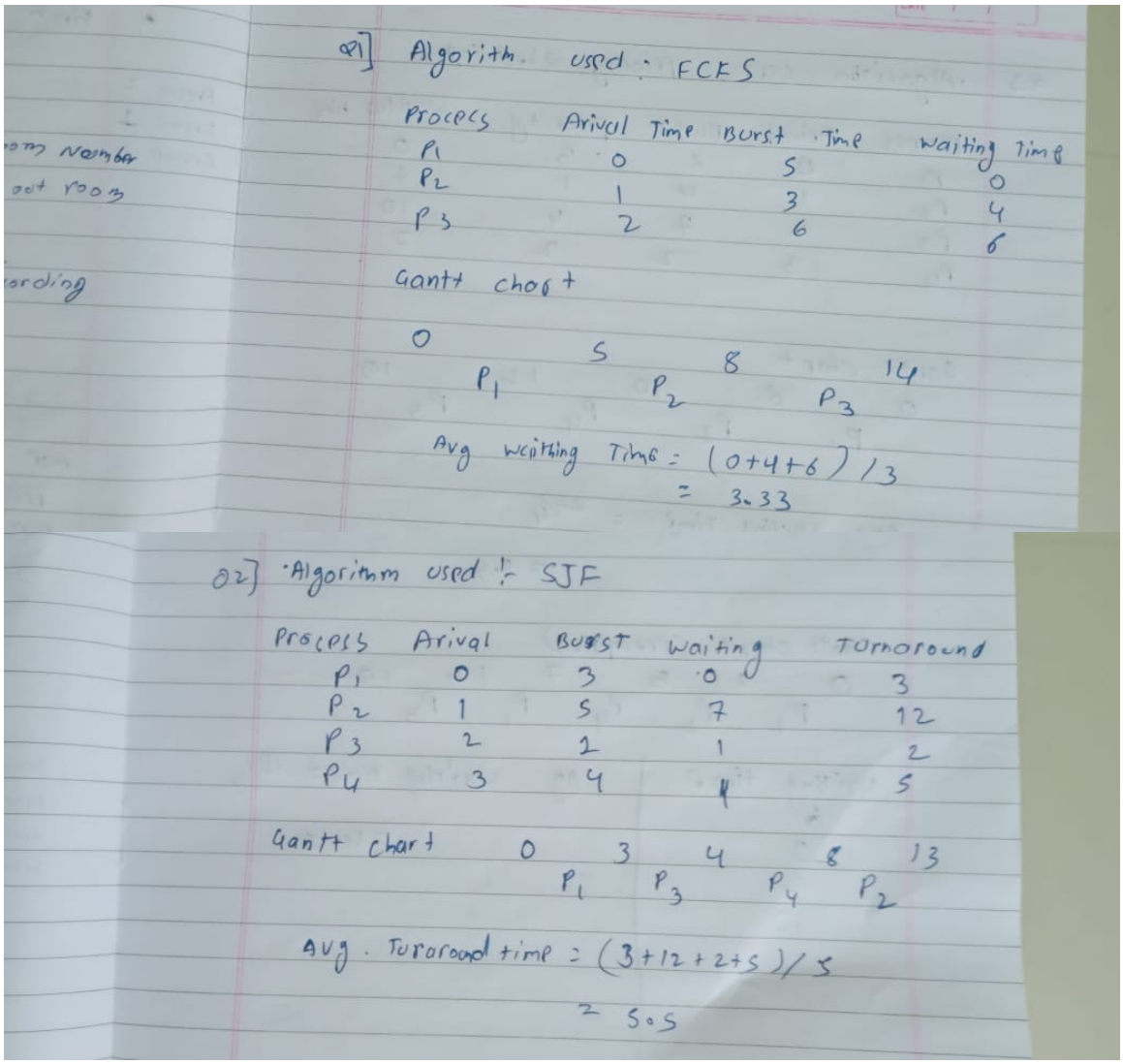
```
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$ nano Assin.sh
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$ cat Assin.sh
echo "Enter a number" ; read a
if [ $a -gt 10 ]
then
       echo "$a is greater than 10"
else
       if [ $a -eq 10 ]
       then
                 echo "$a is equl to 10 "
       else
                 echo "$a is smaller than 10"
fi


cdac@DESKTOP-R3VB19M:~/LinuxAssignment$ ./Assin.sh
Enter a number
6
./Assin.sh: line 14: syntax error: unexpected end of fi
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$ nano Assin.sh
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$ ./Assin.sh
Enter a number
6
6 is smaller than 10
cdac@DESKTOP-R3VB19M:~/LinuxAssignment$
```

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.

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.

Part E



Q1] Algorithm used : FCFS

Process	Arival Time	Burst Time	waiting Time
P_1	0	5	0
P_2	1	3	4
P_3	2	6	6

Gantt chart

0 — P_1 — 5 — P_2 — 8 — P_3 — 14

Avg waiting Time = (0+4+6) / 3
= 3.33

Q2] Algorithm used :- SJF

Process	Arival	Burst	waiting	Turnaround
P_1	0	3	0	3
P_2	1	5	7	12
P_3	2	2	1	2
P_4	3	4	4	5

Gantt chart 0 — P_1 — 3 — P_3 — 4 — P_4 — 8 — P_2 — 13

Avg. Turaround time = (3+12+2+5) / 5
= 5.5

Q3 Algorithm used : Priority Scheduling

Process	Arival	Burst	priority	waiting time
P_1	0	6	3	0
P_2	1	4	1	5
P_3	2	7	4	10
P_4	3	2	2	7

Gantt chart

0 — 6 — 10 — 12 — 19

P_1 P_2 P_4 P_3

Avg Waiting Time = $22/4$

$= 5.5$

Gantt chart (Preemptive)

0 — 1 — 5 — 7 — 12 — 19

P_1 P_2 P_4 P_1 P_3

waiting time

6

0

2

10

Avg. Waiting time = $18/4$

$= 4.5$

Q4] Algorithm used : Round Robin
Quantum = 2 ms

Process	Arrival	Burst	waiting	Turnaround
P_1	0	4	6	10
P_2	1	5	8	13
P_3	2	2	2	4
P_4	3	3	7	10

Gantt chart :- (CPU not kept idle)

0 2 4 6 8 10 12 13 14
P_1 P_2 P_3 P_4 P_1 P_2 P_4 P_2

Avg Turnaround time : (10+13+4+10)/4
= 9.25