Concepts of Operating System
Assignment 2

Part A

What will the following commands do?

1. echo "Hello, World!"
   Ans:
   This command will print Hello World.

2. Name= “Productive”
   Ans:
   The Productive value is assigning to a variable called Name.

3. touch file.txt Ans:
   touch command helps to create a new file called file.txt

4. ls -a Ans:
   It will list all files and directories, including hidden ones, within the current directory.

5. rm file.txt Ans:
   The file.txt will get removed/deleted.

6. cp file1.txt file2.txt Ans:
   The content of file1.txt file will get copied and paste to the destination file file2.txt

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

The mv file.txt /path/to/directory/ command is used move a file from its current location to a specified destination directory.

8. chmod 755 script.sh
   Ans:
   chmod is a command which change the permission of a file. 755 is a octal number that represents the new permissions. here 1st number(7) gives all permission (read,write, execute) to Owner/user. 2nd and 3rd number (5 and 5) gives read and execute permission to group and other users.

9. grep “pattern” file.txt
   Ans:
   grep is command is use to search for a content in a file. It will search for “pattern” word in file1.txt.

10. kill PID Ans:
    The command kill PID is used to terminate a process.

11. mkdir mydir && cd
    mydir && touch file.txt
    && echo "Hello,
    World!" > file.txt && cat
    file.txt Ans:
    This command creates a new directory named mydir, changes the current directory to mydir, creates an empty file named file.txt, writes the text "Hello, World!" into file.txt, and then displays the content of file.txt to the terminal.

12. ls -l | grep ".txt" Ans:
    Lists all files and directories in the current folder and finds all the lines that end in .txt.

13. cat file1.txt file2.txt |
    sort | uniq Ans:
    Combines the contents of file1.txt and file2.txt, sorts all the lines, and then removes any duplicate lines.

14. chmod 644 file.txt Ans:
    Changes the file permissions so the owner can read and write, while others can only read.

15. cp -r source_directory
    destination_directory
    Ans:
    Copies an entire directory and all of its contents to a new location.

16. find /path/to/search -
    name "*.txt" Ans:
    Searches for all files ending with .txt in a specified directory and its subdirectories.

17. chmod u+x file.txt Ans:
    Adds the execute permission to a file for its owner, making the file runnable as a program.

18. Echo $path Ans:
    Prints the list of directories where the system looks for executable commands.

Part B

Identify True or False:

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

2. mv is used to move files and directories. Ans: True

3. cd is used to copy files and directories. Ans: False

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: False

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

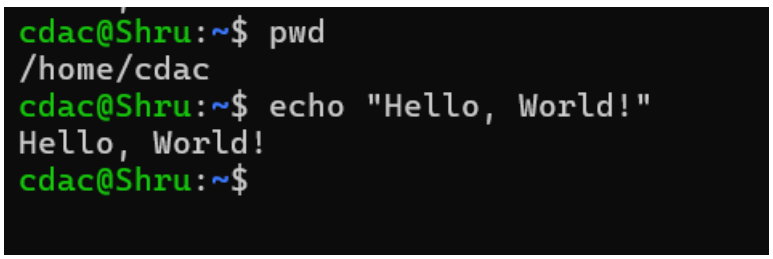
8. rm -rf file.txt deletes a file forcefully without confirmation. Ans: True

Part C

Question 1:

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

Ans:



Question 2:

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

Ans:

```
cdac@Shru:~$ cat print.sh

#!/bin/bash
name="CDAC Mumbai"
echo $name


cdac@Shru:~$ ./print.sh
CDAC Mumbai
cdac@Shru:~$
```

Question 3:

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

Ans:

```
cdac@Shru:~$ vi input.sh
cdac@Shru:~$ chmod +x input.sh
cdac@Shru:~$ ./input.sh
Enter a number:
100
Entered number is 100
cdac@Shru:~$ cat input.sh
#!/bin/bash

echo "Enter a number: "
read Number
echo "Entered number is $Number"

cdac@Shru:~$
```

Question 4:

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

Ans:

```
cdac@Shru:~$ cat add2.sh
#!/bin/bash
echo "Enter two numbers: "
read num1
read num2
sum=$(( num1+num2 ))
echo "Sum is: $sum"



cdac@Shru:~$ ./add2.sh
Enter two numbers:
7
15
Sum is: 22
```

Question 5:

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

Ans:

```
cdac@Shru:~$ cat evenodd.sh
#!/bin/bash
echo "Enter the number : "
read num
if ((num % 2 == 0))
then
        echo "Even!"
else
        echo "Odd"
fi



cdac@Shru:~$ ./evenodd.sh
Enter the number :
80
Even!
```

```
cdac@Shru:~$ ./evenodd.sh
Enter the number :
21
Odd
cdac@Shru:~$
```

Question 6:

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

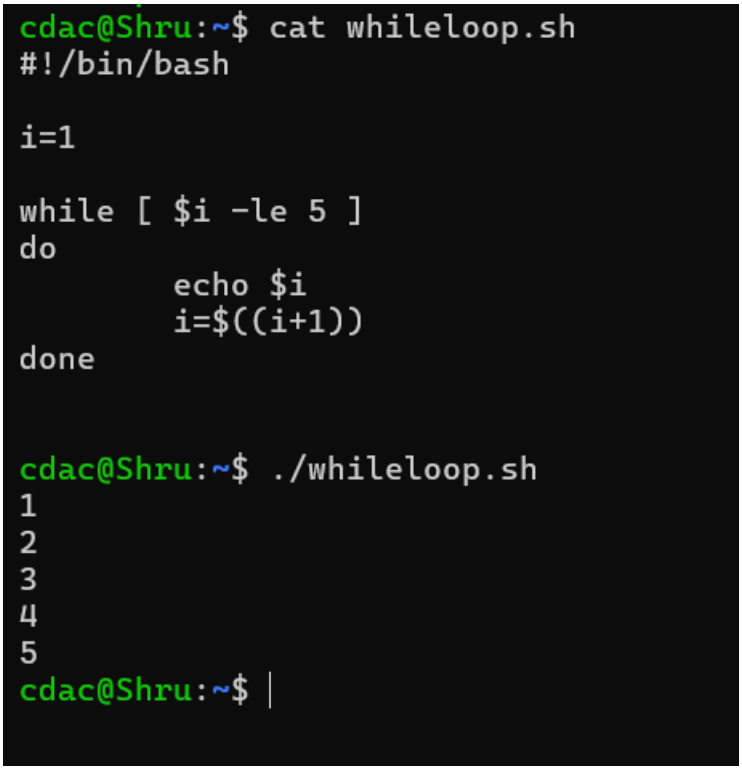
Ans:

```
cdac@Shru:~$ cat forloop.sh
#!/bin/bash
echo "For Loop: "

for (( i=1;i<=5;i++ ))

do
    echo $i
done



cdac@Shru:~$ ./forloop.sh
For Loop:
1
2
3
4
5
cdac@Shru:~$
```

Question 7:

Write a shell script that uses a while loop to print numbers from 1 to 5

Ans:



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".

Ans:

```
#!/bin/bash

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


cdac@Shru:~$ ./chkfleexst.sh
File does not exist
cdac@Shru:~$ touch file.txt
cdac@Shru:~$ ./chkfleexst.sh
File exists
cdac@Shru:~$
```

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.

Ans:

```
cdac@Shru:~$ vi greaterthan.sh
cdac@Shru:~$ chmod +x greaterthan.sh
cdac@Shru:~$ ./greaterthan.sh
Enter a number:
8
Number is not greater than 10!
cdac@Shru:~$ ./greaterthan.sh
Enter a number:
18
Number is greater than 10!
cdac@Shru:~$ cat greaterthan.sh
#!/bin/bash

echo "Enter a number: "
read number

if [ "$number" -gt 10 ]
then
        echo "Number is greater than 10!"
else
        echo "Number is not greater than 10!"
fi
```

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.

Ans:

```
cdac@Shru:~$ vi multiplication.sh
cdac@Shru:~$ chmod +x multiplication.sh
cdac@Shru:~$ ./multiplication.sh
Multiplication table of 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@Shru:~$ cat multiplication.sh
#!/bin/bash

echo "Multiplication table of 1 to 5 : "
for i in {1..5}
do
        for j in {1..5}
        do
                result=$((i*j))
                echo -ne "$result\t"
        done
        echo ""
done


cdac@Shru:~$
```

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.

Ans:

```
cdac@Shru:~$ vi whileloopQ11.sh
cdac@Shru:~$ chmod +x whileloopQ11.sh
cdac@Shru:~$ ./whileloopQ11.sh
Enter a positive number ( or negative number to exit):
2
Square of 2 is : 4
Enter a positive number ( or negative number to exit):
7
Square of 7 is : 49
Enter a positive number ( or negative number to exit):
-2
Negative number entered. Thank you...
Script finished......
cdac@Shru:~$ cat whileloopQ11.sh
#!/bin/bash

while true
do
        echo "Enter a positive number ( or negative number to exit): "
        read number
        if [ "$number" -lt 0 ]
        then
                echo "Negative number entered. Thank you..."
                break
        fi
        square=$((number*number))
        echo "Square of $number is : $square"
done
echo "Script finished......"
```

Part D

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.

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.

---

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.

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.

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:

Date: / /

Assignment 2:
OS.

Q.1. Consider following process with arrival time & burst time.

Process	Arrival time	Burst time
P1	0	5
P2	1	3
P3	2	6

Calculate avg. waiting time using FCFS scheduling.

⇒ P1 : WT = 0 - 0 = 0
P2 : WT = 5 - 1 = 4
P3 : WT = 8 - 2 = 6

P1 = 5	P2 = 3	P3 = 6
0	5	8	14

Avg. waiting time = Total WT / 3

= 10/3 = 3.33.

Q.2 Consider data:

Process	Arrival time	Burst time
P1	0	3
P2	1	5
P3	2	1
P4	3	4

Calculate avg. turn around time using SJF scheduling.

⇒

P1 = 3	P3 = 1	P4 = 4	P5 = 5
0	3	4	8	13

Date: / /

P1 P1 = TT = 3-0 = 3
P2 P3 = TT = 4-2 = 2
P. P4 = TT = 8-3 = 5
P2 = TT = 13-1 = 12
Total TT = 22
∴ avg TT = 22/4 = 5.5

Q.3. Consider data.

Process	AT	BT	Priority
P1	0	6	3
P2	1	4	1
P3	2	7	4
P4	3	2	2

Calculate avg. waiting time using priority scheduling.

⇒

P1 = waiting time = (1-0) + (5-1)
P2 = WT = (5-1) + (7-3) = 6 ⌋ = 4.
P3 = WT = 7-2 = 5
P4 = WT = 3-3 = 0
Total avg. waiting time = 15
avg = 15/4
= 3.75

P1 = 6	P2 = 4	P4 = 2	P3 = 7
0	6	10	12	19

Q.4. Consider table:

Process	Arrival time	Burst time
P1	0	4
P2	1	5
P3	2	2
P4	3	3

Date: / /

- Calculate avg turn around time using Round Robin scheduling. is 2 units.

⇒ P1 = Turn around = 10-0=10
P2 = TT = 14-1 = 13
P3 = TT = 6-2 = 4
P4 = TT = 13-3 = 10

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

Total = 37
∴ avg = $\frac{37}{4}$ = 9.25

Q.5. Consider program that uses fork() system call to create child process. Initially parent process has variable x with value of 5. After forking, both parent & child processes increment ~~be~~ value of x by 1. What will be final values of x in parent & child processes after fork() call?

⇒ step 1] Before fork()
parent process has variable
x = 5.

step 2] fork() call.
fork() creates child process, which is almost exact copy of parent.
∴ x copied, not shared.
parent x = 5
child x = 5

Date: / /

step3] Increment in both processes

parent x = 6
child x = 6

∴ x = 6 for independent parent and child values.