Assignment – 2

COS

PART – A

What will the following commands do?

- **echo “Hello, World!”**

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

- **name=”Productive”**

  It assigns the string productive to name variable which can be printed by echo.

```
cdac@LAPTOP-BFJCJVNG: ~
cdac@LAPTOP-BFJCJVNG:~$ name="Productive"
cdac@LAPTOP-BFJCJVNG:~$ echo $name
Productive
cdac@LAPTOP-BFJCJVNG:~$
```

- **touch file.txt**

  It creates a file named file.txt on which we can write our instructions.

```
cdac@LAPTOP-BFJCJVNG: ~
cdac@LAPTOP-BFJCJVNG:~$ touch file.txt
cdac@LAPTOP-BFJCJVNG:~$ ls
LinuxAssignment  file.txt
cdac@LAPTOP-BFJCJVNG:~$
```

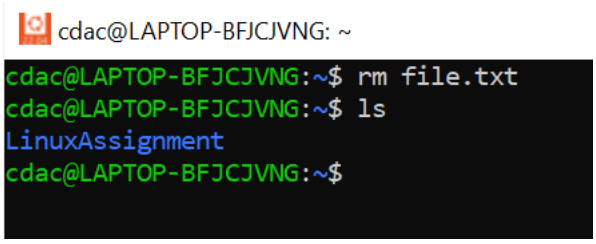
- **ls -a**

  It lists all files and directories in the current directory, including hidden files.

```
cdac@LAPTOP-BFJCJVNG: ~
cdac@LAPTOP-BFJCJVNG:~$ ls -a
.   .bash_history  .bashrc     .local       .profile                 LinuxAssignment
..  .bash_logout   .landscape  .motd_shown  .sudo_as_admin_successful  file.txt
cdac@LAPTOP-BFJCJVNG:~$
```

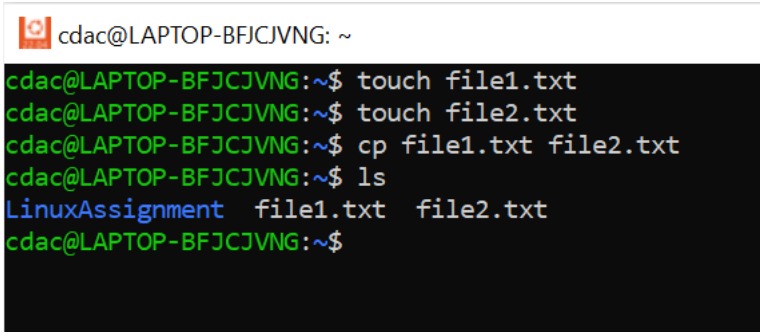
- **rm file.txt**

  It removes the file named file.txt



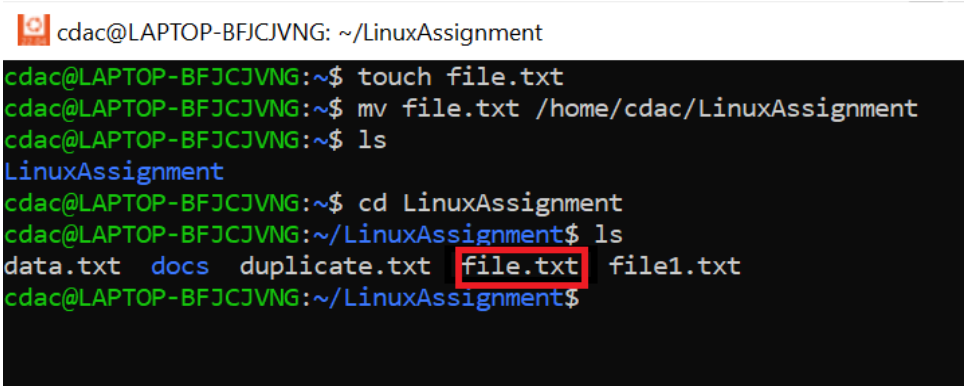
- **cp file1.txt file2.txt**

  It copies the contents of file 1 into file 2.



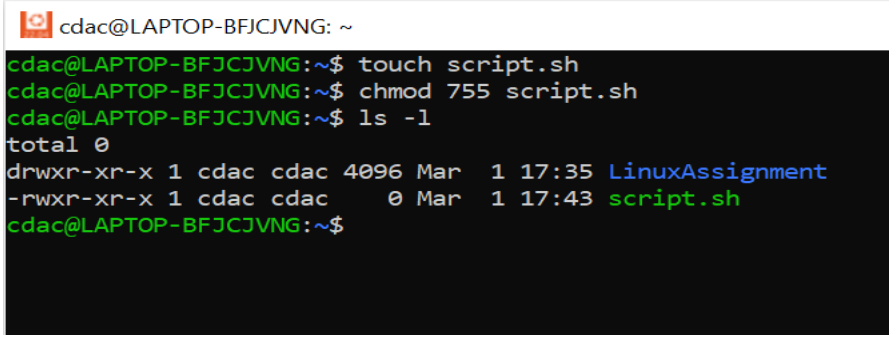
- **mv file.txt /path/to/directory/**

  It moves a file to a particular directory.

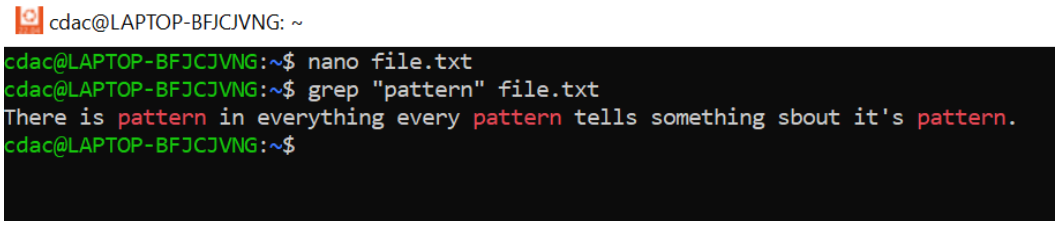


- **chmod 755 script.sh**

  It gives read, write, execute permission to the main user and only read and execute permission to group and other user with respect to script.sh file.

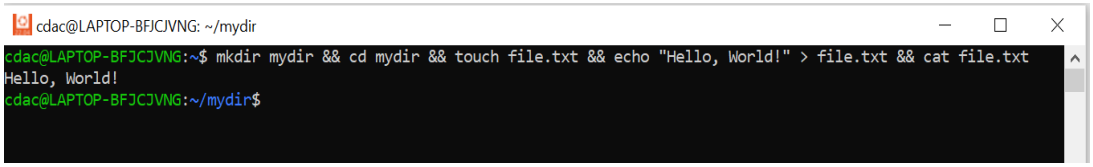


- **grep “pattern” file.txt**
  It searches for pattern word inside the file.txt

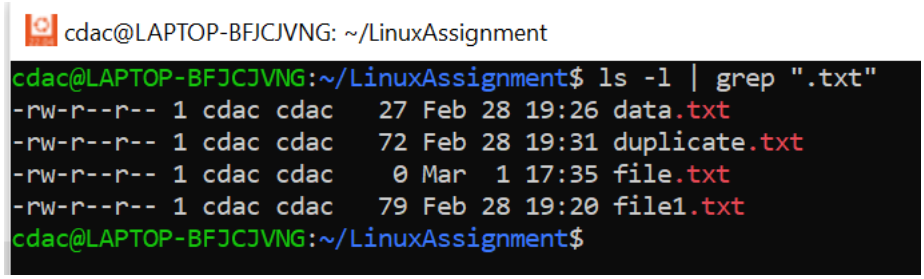


- **kill PID**
  It is used to terminate a process with a specific Process ID (PID).

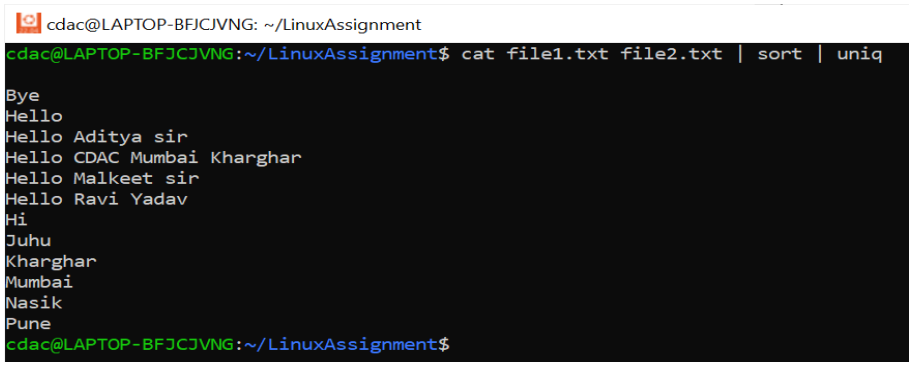
- **mkdir mydir && cd mydir && touch file.txt && echo “Hello, World!” > file.txt && cat file.txt**
  Creates a new directory named mydir, moves into newly created directory, creates an empty file named file.txt, writes “Hello, World!” into file.txt, displays the content of file.txt in the terminal.



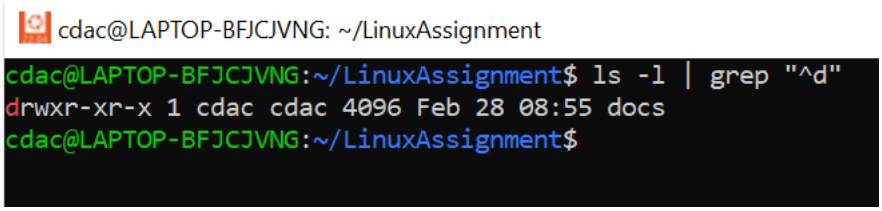
- **ls -l | grep “.txt”**
  It lists all .txt files in the current directory with detailed information



- **cat file1.txt file2.txt | sort | uniq**
  It sorts the file1.txt and file2.txt by clubbing all the duplications and uniq will remove the duplications.

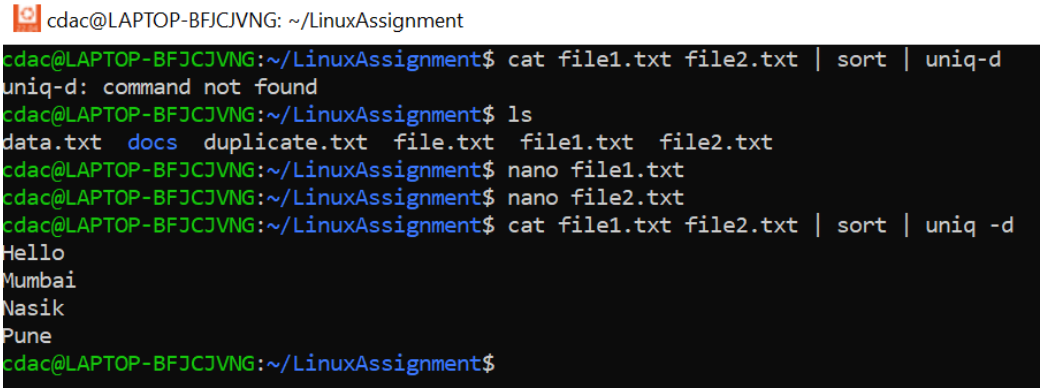


- **ls -l | grep "^d"**
  It is used to show current directories with the user.

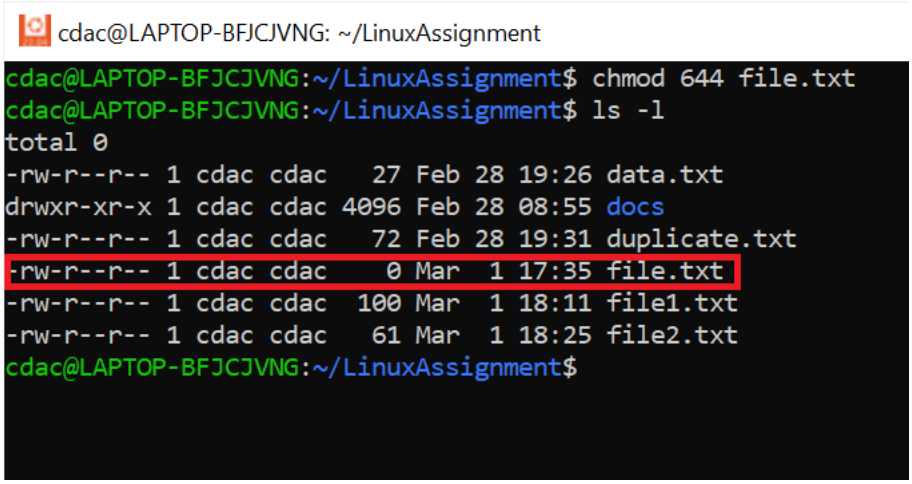


- **grep -r "pattern" /path/to/directory/**
  It is used to search for a specific pattern inside all files within a given directory and its subdirectories.

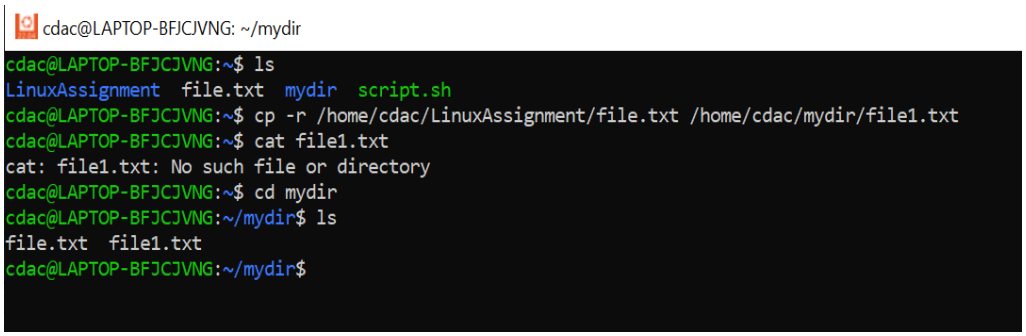
- **cat file1.txt file2.txt | sort | uniq –d**
  is used to find duplicate lines that appear in both file1.txt and file2.txt



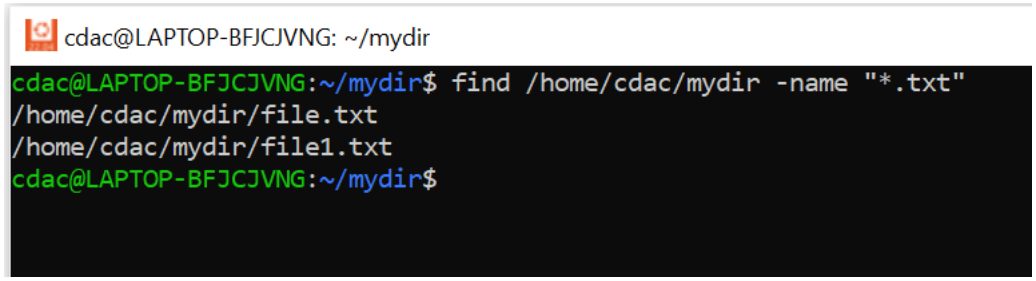
- **chmod 644 file.txt**
  It changes the file permission of the user to read and write, and of group and other user to only read.



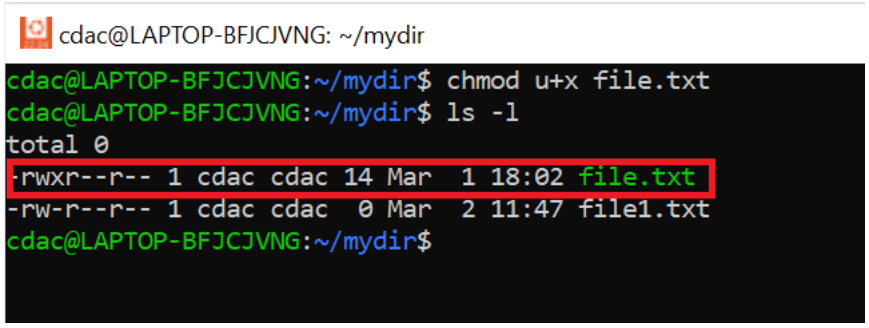
- **cp -r source_directory destination_directory**
  It is used to copy a directory and its contents (including subdirectories and files) to another location in Linux.



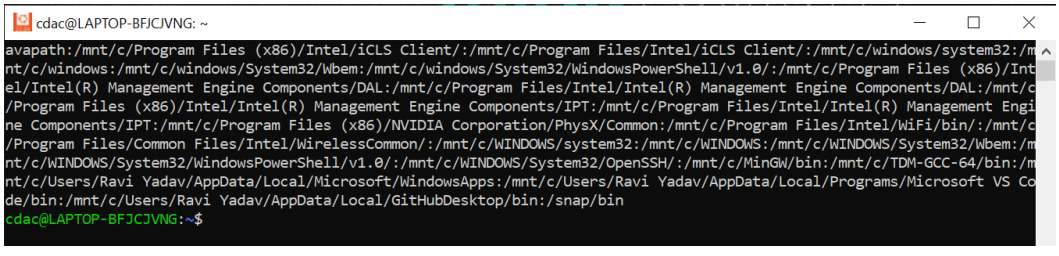
- **find /path/to/search -name "*.txt"**
  It will find .txt file in the given directory.



- **chmod u+x file.txt**
  It gives user the permission to execute with respect to file.txt.



- **echo $PATH**
  It will print the environment variable that stores a list of directories where executable files (commands, scripts, programs) are located.



PART – B

Identify True or False:

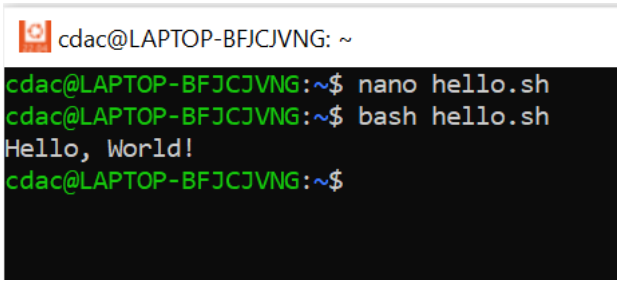
1. **ls** is used to list files and directories in a directory. (True)
2. **mv** is used to move files and directories. (True)
3. **cd** is used to copy files and directories. (True)
4. **pwd** stands for "print working directory" and displays the current directory. (True)
5. **grep** is used to search 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:

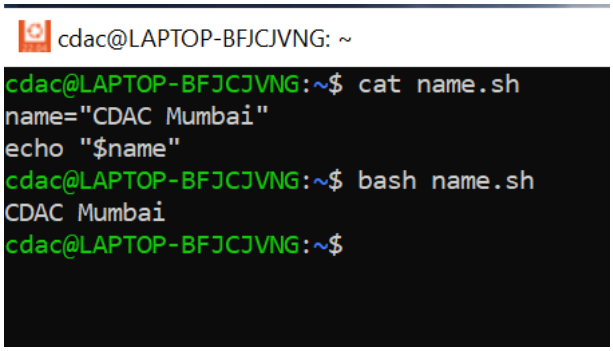
1. **chmodx** is used to change file permissions. (Incorrect) → chmod (Correct)
2. **cpy** is used to copy files and directories. (Incorrect) → cp (Correct)
3. **mkfile** is used to create a new file. (Correct)
4. **catx** is used to concatenate files. (Incorrect) → cat (Correct)
5. **rn** is used to rename files. (Correct)

PART – C

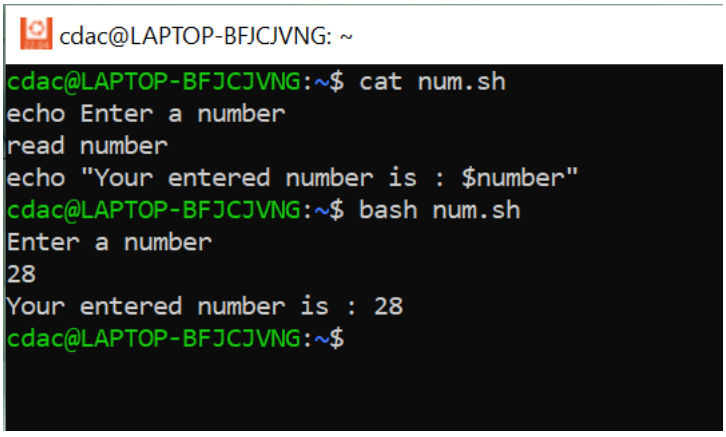
Question 1: Write a shell script that prints "Hello, World!" to the terminal.



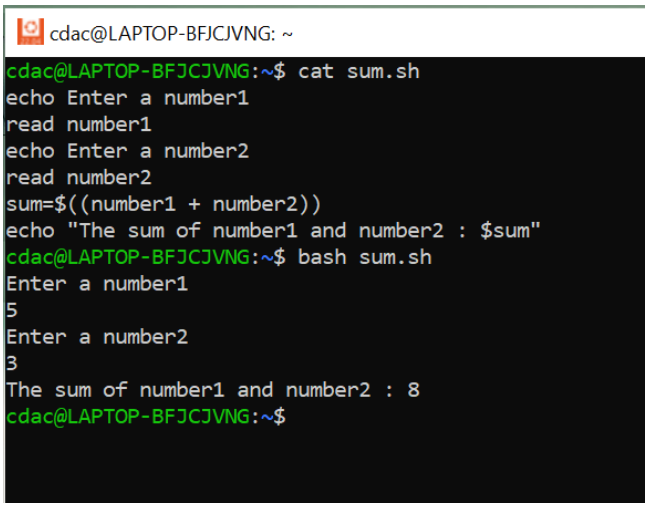
Question 2: Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable.



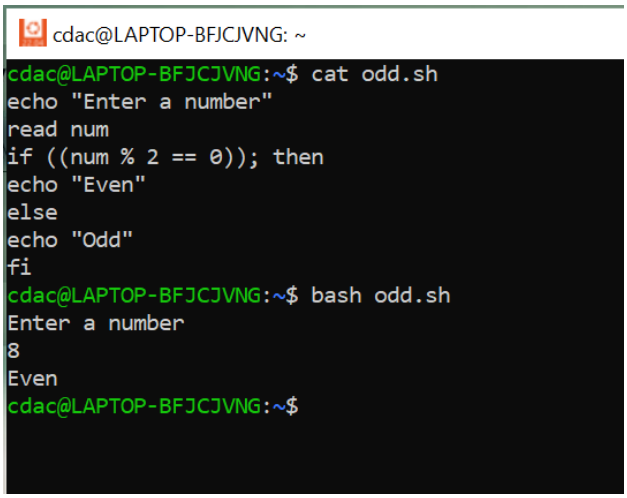
Question 3: Write a shell script that takes a number as input from the user and prints it.



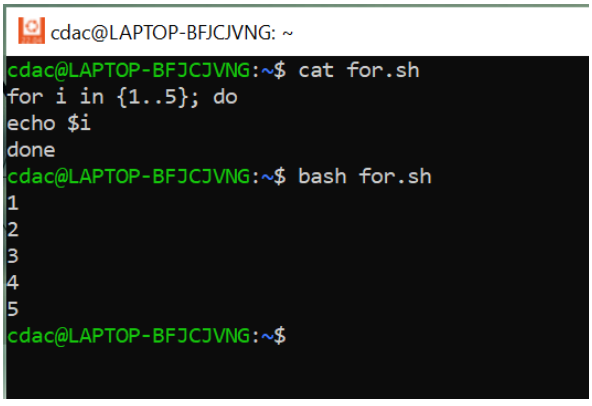
Question 4: Write a shell script that performs addition of two numbers (e.g., 5 and 3) and prints the result.



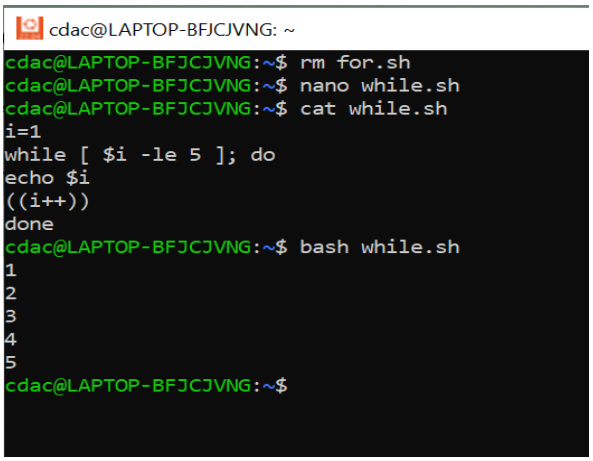
Question 5: Write a shell script that takes a number as input and prints "Even" if it is even, otherwise prints "Odd".



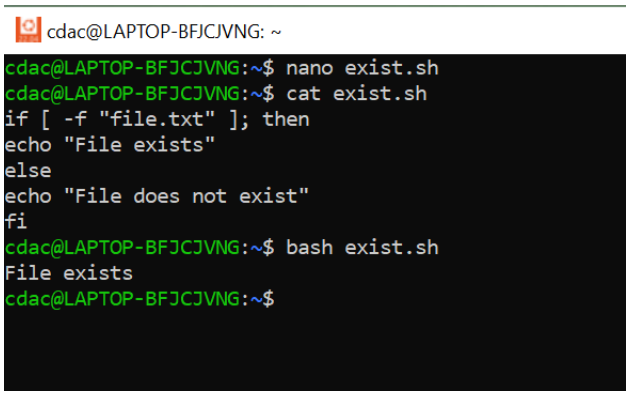
Question 6: Write a shell script that uses a for loop to print numbers from 1 to 5.



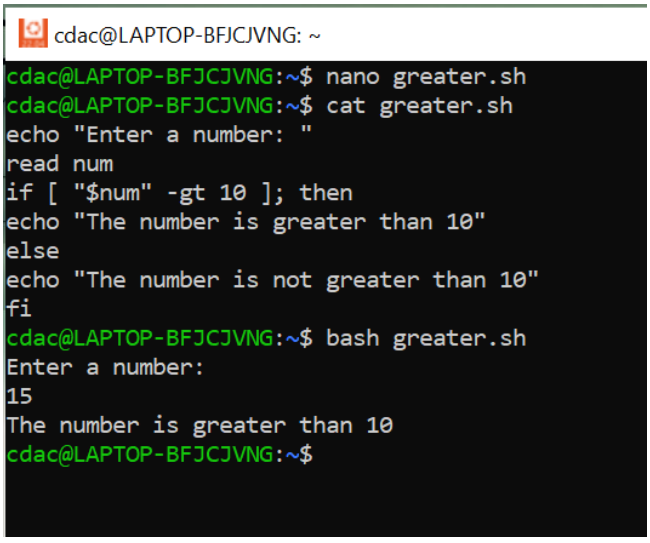
Question 7: Write a shell script that uses a while loop to print numbers from 1 to 5.



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".



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.



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@LAPTOP-BFJCJVNG: ~
cdac@LAPTOP-BFJCJVNG:~$ nano multi.sh
cdac@LAPTOP-BFJCJVNG:~$ bash multi.sh
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
   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-BFJCJVNG:~$
```

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@LAPTOP-BFJCJVNG: ~
cdac@LAPTOP-BFJCJVNG:~$ nano last.sh
cdac@LAPTOP-BFJCJVNG:~$ cat last.sh
while true; do
echo "Enter a number (negative to exit);"
read num
if [ "$num" -lt 0 ]; then
echo "Negative number entered. Exiting.."
break
fi
square=$((num * num))
echo "Square of $num is: $square"
done
cdac@LAPTOP-BFJCJVNG:~$ bash last.sh
Enter a number (negative to exit);
3
Square of 3 is: 9
Enter a number (negative to exit);
5
Square of 5 is: 25
Enter a number (negative to exit);
8
Square of 8 is: 64
Enter a number (negative to exit);
-2
Negative number entered. Exiting..
cdac@LAPTOP-BFJCJVNG:~$
```

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.

Answer 1) Avg. Waiting Time = 3.33.

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.

Answer 2) Avg Turnaround Time = 5.5.

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.

Answer 3) Avg. Waiting Time = 7.75.

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.

Answer 4) Avg. Turnaround time = 9.

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?

Answer 5)

Let x = 5.

Call fork().

Both parent and child increase x by 1.

child x = 6.

parent x = 6.

Final values of x:

child process, x = 6.

parent process, x = 6.