Name : Sohail Khan

CDAC MUMBAI

Concepts of Operating System Assignment 2

Part A

What will the following commands do?

- echo "Hello, World!"
  ⇨ Prints "Hello, World!"
- name="Productive"
  ⇨ Creates a variable (name) and stores a string value ("Productive")
- touch file.txt
  ⇨ Creates a file named file.txt
- ls -a
  ⇨ ls lists all the files and directories while -a also lists the hidden ones
- rm file.txt
  ⇨ removes file.txt
- cp file1.txt file2.txt
  ⇨ Copies file1.txt content in file2.txt
- mv file.txt /path/to/directory/
  ⇨ It will move the file.txt file into new directory according to the path
- chmod 755 script.sh
  ⇨ It will change permissions: users get all, group and others get read and execute only.
- grep "pattern" file.txt
  ⇨ Search for the word “pattern” in file.txt
- kill PID
  ⇨ kills the process w.r.to the Process ID.
- mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.tx
  ⇨ 1)Create mydir 2) Enter mydir 3) Create file.txt 4) Write "Hello, World" in it 5) Print its content.
- ls -l | grep ".txt"
  ⇨ It will search for ".txt" and list all the matched files and directories.
- cat file1.txt file2.txt | sort | uniq
  ⇨ First it will sort the combined output then Prints all the unique values from file1.txt and file2.txt.
- ls -l | grep "^d"
  ⇨ lists only the directories in the current directory.
- grep -r "pattern" /path/to/directory/
  ⇨ Search for the “pattern” word recursively in all the files present in the given path.

- cat file1.txt file2.txt | sort | uniq –d
  ⇨ displays duplicate lines that appear in both file1.txt and file2.txt.
- chmod 644 file.txt
  ⇨ Change the permissions of file.txt : users get read and write, group and others get read only.
- cp -r source_directory destination_directory
  ⇨ Copies the entire source_directory and its contents recursively to destination_directory.
- find /path/to/search -name "*.txt"
  ⇨ Searches for all files with a ".txt" extension in the specified directory and its subdirectories.
- chmod u+x file.txt
  ⇨ changes the permissions of file.txt by adding execute permission for the user (owner) of the file.
- echo $PATH
  ⇨ displays the current value of the 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.
- **TRUE**

Identify the Incorrect Commands:
1. chmodx is used to change file permissions.

- **CORRECT COMMAND** => ***chmod***

2. cpy is used to copy files and directories.

- **CORRECT COMMAND** => ***cp***

3. mkfile is used to create a new file.

- **CORRECT COMMAND** => ***touch <filename>***

4. catx is used to concatenate files.

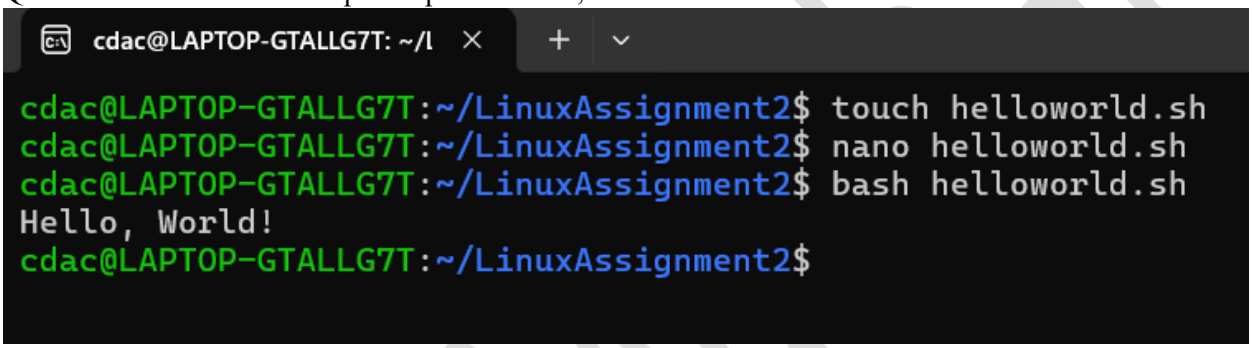
- **CORRECT COMMAND** => ***cat***

5. rn is used to concatenate files

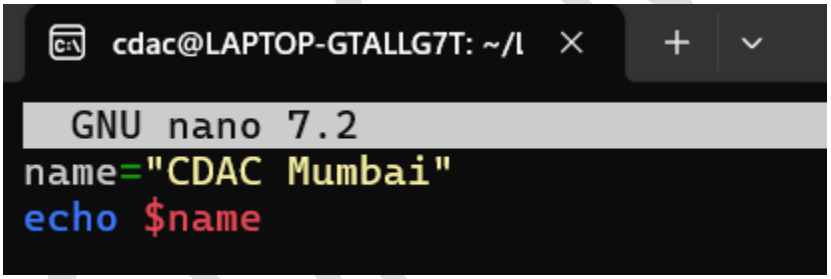
- **CORRECT COMMAND** => ***mv***

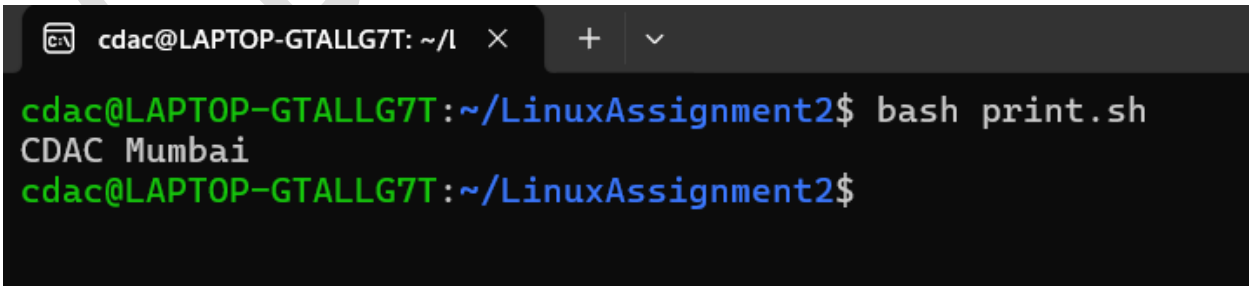
Part C

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

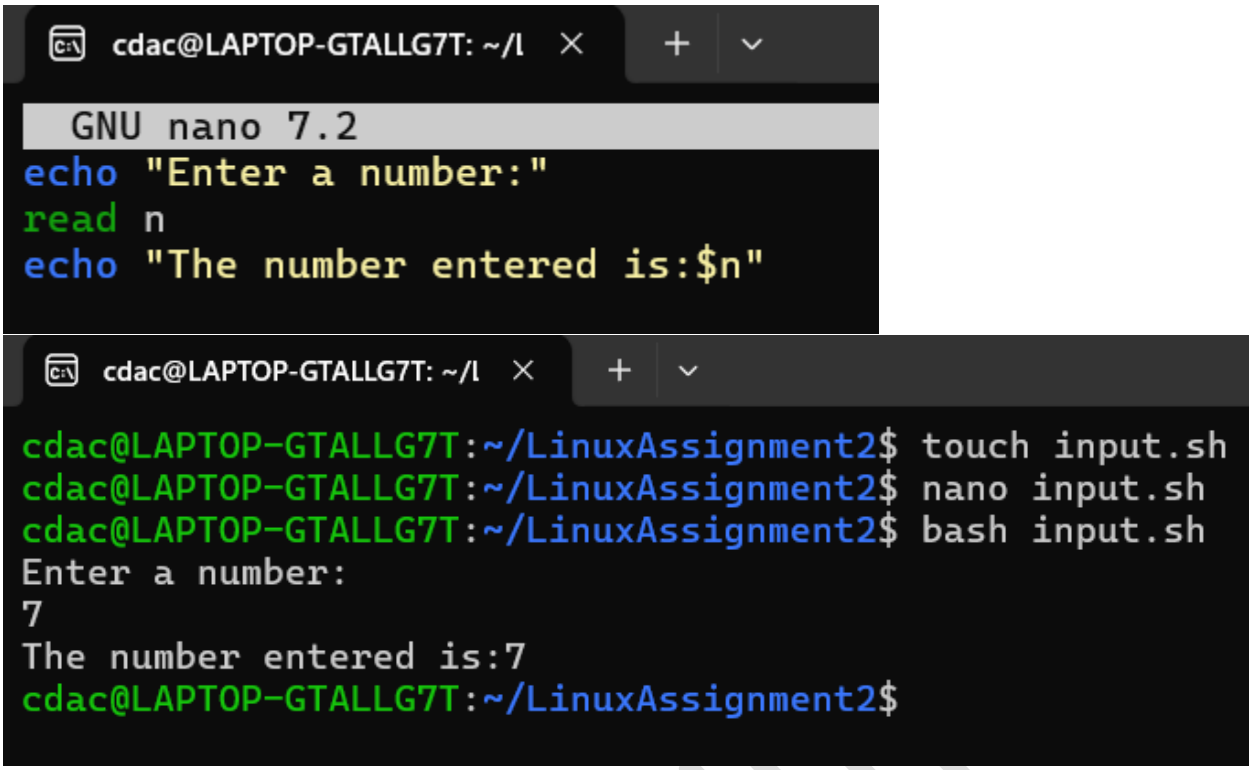


Question 2: Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable.

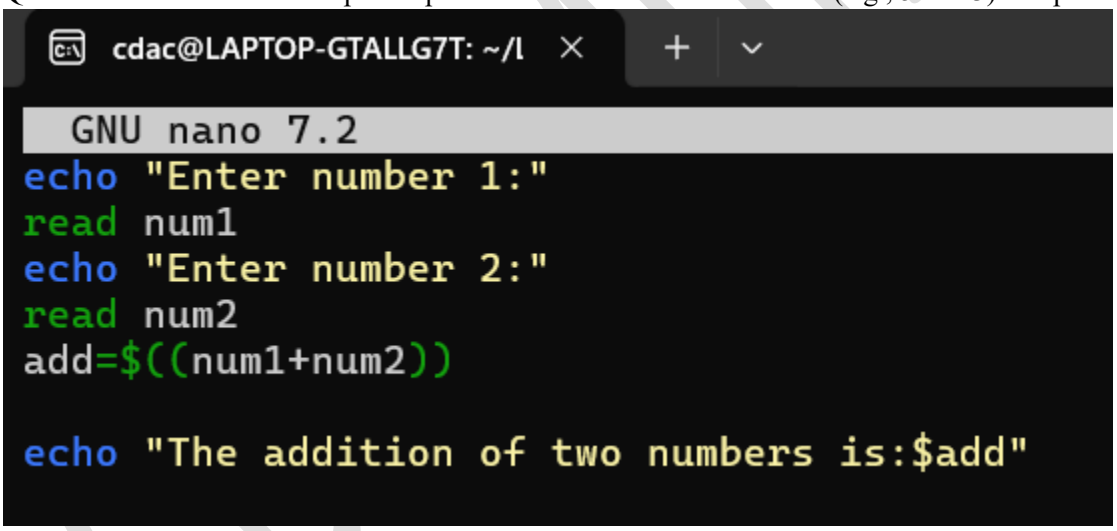




Question 3: Write a shell script that takes a number as input from the user and prints it.



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



```
cdac@LAPTOP-GTALLG7T: ~/l
cdac@LAPTOP-GTALLG7T:~/LinuxAssignment2$ nano add.sh
cdac@LAPTOP-GTALLG7T:~/LinuxAssignment2$ bash add.sh
Enter number 1:
5
Enter number 2:
3
The addition of two numbers is:8
cdac@LAPTOP-GTALLG7T:~/LinuxAssignment2$
```

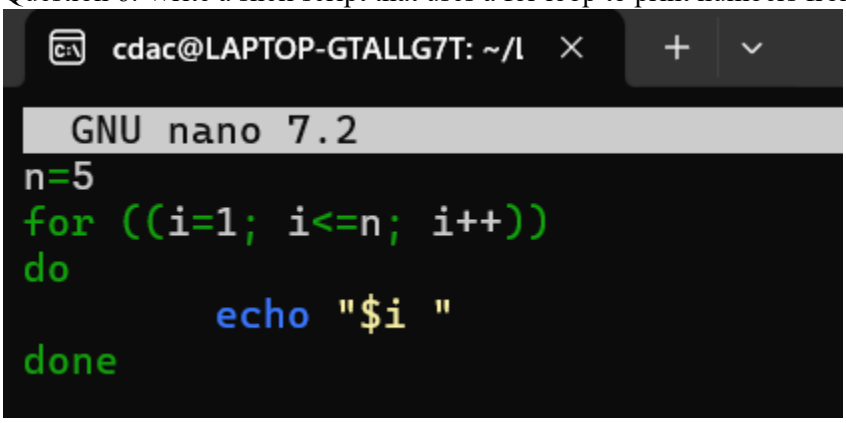
Question 5: Write a shell script that takes a number as input and prints "Even" if it is even, otherwise prints "Odd".

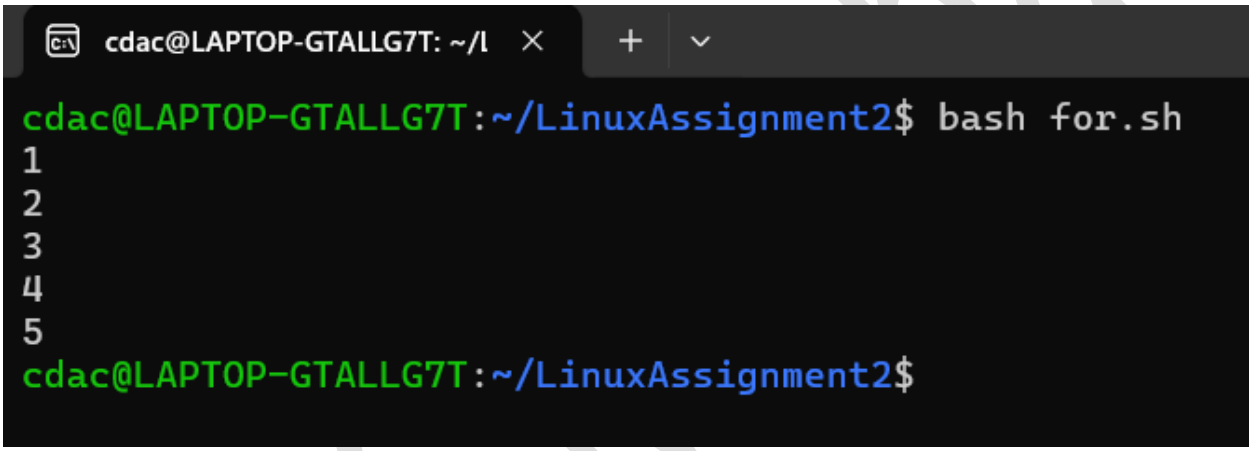
```
cdac@LAPTOP-GTALLG7T: ~/l
  GNU nano 7.2
echo "Enter a number:"
read num

if ((num%2==0)); then
        echo "Even"
else
        echo "Odd"
fi
```

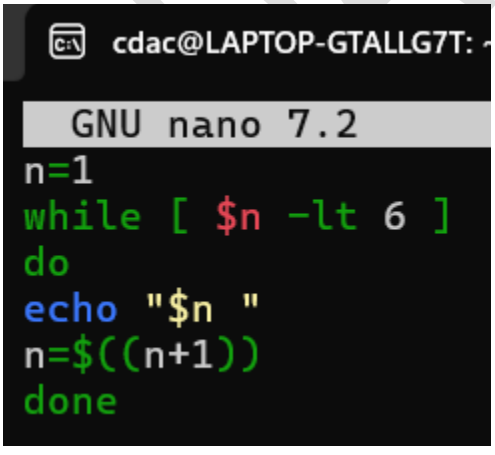
```
cdac@LAPTOP-GTALLG7T: ~/l
cdac@LAPTOP-GTALLG7T:~/LinuxAssignment2$ touch oddeven.sh
cdac@LAPTOP-GTALLG7T:~/LinuxAssignment2$ nano oddeven.sh
cdac@LAPTOP-GTALLG7T:~/LinuxAssignment2$ bash oddeven.sh
Enter a number:
3
Odd
cdac@LAPTOP-GTALLG7T:~/LinuxAssignment2$
```

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.



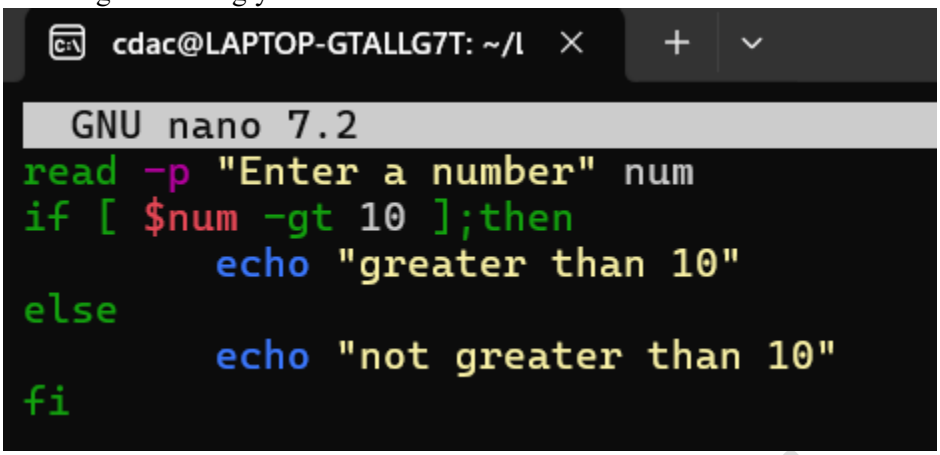
```
cdac@LAPTOP-GTALLG7T: ~/l
cdac@LAPTOP-GTALLG7T:~/LinuxAssignment2$ touch while.sh
cdac@LAPTOP-GTALLG7T:~/LinuxAssignment2$ nano while.sh
cdac@LAPTOP-GTALLG7T:~/LinuxAssignment2$ bash while.sh
1
2
3
4
5
cdac@LAPTOP-GTALLG7T:~/LinuxAssignment2$
```

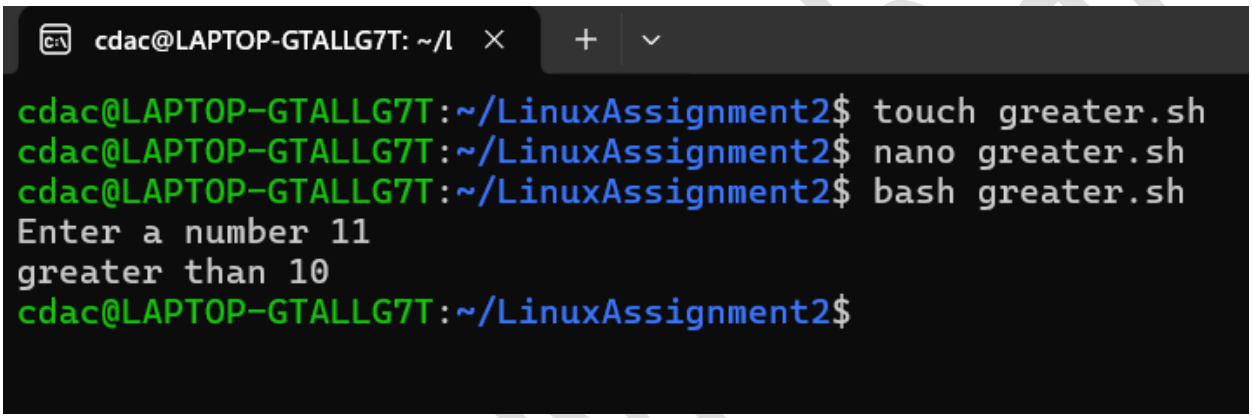
Question 8: Write a shell script that checks if a file named "file.txt" exists in the current directory. If it does, print "File exists", otherwise, print "File does not exist".

```
cdac@LAPTOP-GTALLG7T: ~/l
  GNU nano 7.2
if [ -f "file.txt" ]; then
        echo "file exists"
else
        echo "does not exist"
fi
```

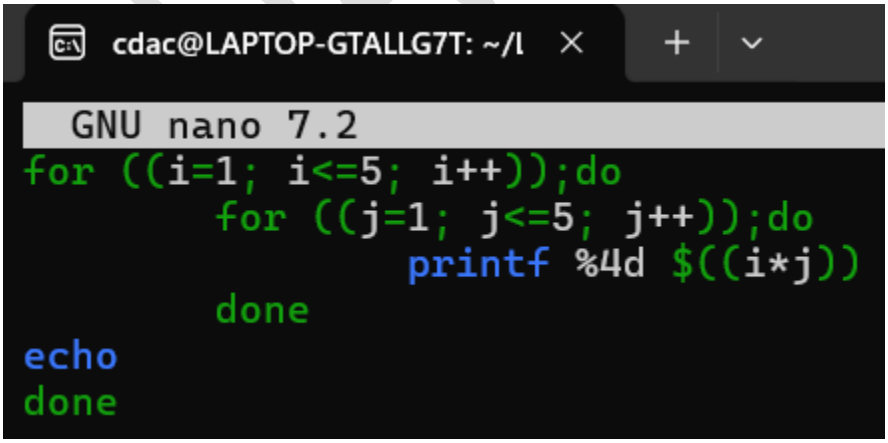
```
cdac@LAPTOP-GTALLG7T: ~/l
cdac@LAPTOP-GTALLG7T:~/LinuxAssignment2$ bash check.sh
file exists
cdac@LAPTOP-GTALLG7T:~/LinuxAssignment2$ ls
add.sh  check.sh  file.txt  for,sh  for.sh  helloworld.sh  input.sh  oddeven.sh  print.sh  print.txt  while.sh
cdac@LAPTOP-GTALLG7T:~/LinuxAssignment2$
```

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-GTALLG7T: ~/l
cdac@LAPTOP-GTALLG7T:~/LinuxAssignment2$ nano MultiplicationTable.sh
cdac@LAPTOP-GTALLG7T:~/LinuxAssignment2$ bash MultiplicationTable.sh
   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@LAPTOP-GTALLG7T:~/LinuxAssignment2$
```

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-GTALLG7T: ~/l
  GNU nano 7.2
while true;do
        read -p "Enter a number" num
        if ((num<0));then
                break
        fi
        echo "square of $num is $((num*num))"
done
```

```
cdac@LAPTOP-GTALLG7T: ~/l
cdac@LAPTOP-GTALLG7T:~/LinuxAssignment2$ bash negative.sh
Enter a number 10
square of 10 is 100
Enter a number 5
square of 5 is 25
Enter a number 14
square of 14 is 196
Enter a number -3
cdac@LAPTOP-GTALLG7T:~/LinuxAssignment2$
```

Part D

1. What is an operating system, and what are its primary functions?
An operating system manages hardware, software, and resources while providing a user interface.
2. Explain the difference between process and thread.
A process is an executing program, while a thread is a lightweight unit of execution within a process.
3. What is virtual memory, and how does it work?
Virtual memory extends RAM using disk space, allowing more processes to run than physically possible.
4. Describe the difference between multiprogramming, multitasking, and multiprocessing.
Multiprogramming loads multiple programs in memory, multitasking switches tasks, and multiprocessing uses
multiple CPUs.
5. What is a file system, and what are its components?
A file system organizes and manages data storage, including files, directories, and metadata.
6. What is a deadlock, and how can it be prevented?
Deadlock occurs when processes block each other indefinitely; it can be prevented by avoiding circular wait
conditions.
7. Explain the difference between a kernel and a shell.
The kernel is the core of the OS, while the shell is the user interface for interacting with the OS.
8. What is CPU scheduling, and why is it important?
CPU scheduling selects processes for execution to optimize performance and resource utilization.
9. How does a system call work?
A system call allows user programs to request OS services via a controlled interface.
10. What is the purpose of device drivers in an operating system?
Device drivers enable communication between the OS and hardware devices.
11. Explain the role of the page table in virtual memory management.
A page table maps virtual addresses to physical memory in virtual memory management.
12. What is thrashing, and how can it be avoided?
Thrashing happens when excessive paging slows down a system; it can be avoided by adjusting working sets.
13. Describe the concept of a semaphore and its use in synchronization.
A semaphore is a synchronization mechanism that controls process access to shared resources.
14. How does an operating system handle process synchronization?
The OS handles process synchronization using locks, semaphores, and monitors to prevent race conditions.
15. What is the purpose of an interrupt in operating systems?
An interrupt signals the CPU to pause execution and handle urgent tasks.
16. Explain the concept of a file descriptor.
A file descriptor is a unique identifier for an open file or I/O resource.
17. How does a system recover from a system crash?
A system recovers from a crash using logs, backups, and system checkpoints.
18. Describe the difference between a monolithic kernel and a microkernel.
A monolithic kernel has all OS services in one unit, while a microkernel has minimal services in the core.
19. What is the difference between internal and external fragmentation?
Internal fragmentation occurs within allocated memory blocks, while external fragmentation happens between blocks.
20. How does an operating system manage I/O operations?
The OS manages I/O operations using buffering, caching, and device scheduling.

21. Explain the difference between preemptive and non-preemptive scheduling.
Preemptive scheduling allows task interruption, while non-preemptive scheduling completes tasks before
switching.
22. What is round-robin scheduling, and how does it work?
Round-robin scheduling assigns time slices to processes in a cyclic manner.
23. Describe the priority scheduling algorithm. How is priority assigned to processes?
Priority scheduling executes processes based on priority levels, where higher-priority tasks run first.
24. What is the shortest job next (SJN) scheduling algorithm, and when is it used?
SJN executes the shortest job first, minimizing waiting time in batch processing.
25. Explain the concept of multilevel queue scheduling.
Multilevel queue scheduling categorizes processes into priority-based queues with different scheduling policies.
26. What is a process control block (PCB), and what information does it contain?
A PCB stores process-related information like state, program counter, and resources.
27. Describe the process state diagram and the transitions between different process states.
A process transitions between new, ready, running, waiting, and terminated states.
28. How does a process communicate with another process in an operating system?
Processes communicate via inter-process communication (IPC) methods like message passing and shared
memory.
29. What is process synchronization, and why is it important?
Process synchronization ensures orderly execution to prevent race conditions and data inconsistency.
30. Explain the concept of a zombie process and how it is created.
A zombie process has completed execution but still holds an entry in the process table.
31. Describe the difference between internal fragmentation and external fragmentation.
Internal fragmentation wastes memory inside allocated blocks, while external fragmentation wastes unallocated space.
32. What is demand paging, and how does it improve memory management efficiency?
Demand paging loads pages into memory only when needed, improving efficiency.
33. Explain the role of the page table in virtual memory management.
A page table maps virtual addresses to physical frames in memory management.
34. How does a memory management unit (MMU) work?
The MMU translates virtual addresses to physical addresses for efficient memory access.
35. What is thrashing, and how can it be avoided in virtual memory systems?
Thrashing degrades performance due to excessive paging; it is avoided by optimizing working sets.
36. What is a system call, and how does it facilitate communication between user programs and the operating system?
A system call lets programs request OS services like file handling and process management.
37. Describe the difference between a monolithic kernel and a microkernel.
A monolithic kernel has integrated services, while a microkernel has minimal core functionality.
38. How does an operating system handle I/O operations?
The OS manages I/O via interrupt handling, buffering, and device scheduling.
39. Explain the concept of a race condition and how it can be prevented.
A race condition occurs when multiple processes access shared data unpredictably; locking mechanisms
prevent it.
40. Describe the role of device drivers in an operating system.
Device drivers facilitate communication between hardware and the OS.

41. What is a zombie process, and how does it occur? How can a zombie process be prevented?
A zombie process has completed execution but retains an entry; it is prevented by the parent handling termination.
42. Explain the concept of an orphan process. How does an operating system handle orphan processes?
An orphan process loses its parent; the OS reassigns it to the init process.
43. What is the relationship between a parent process and a child process in the context of process management?
A parent creates a child process, which can execute independently or under its parent's control.
44. How does the fork() system call work in creating a new process in Unix-like operating systems?
The fork() system call creates a new process by duplicating the parent process.
45. Describe how a parent process can wait for a child process to finish execution.
A parent process waits for its child using the wait() system call.
46. What is the significance of the exit status of a child process in the wait() system call?
The exit status informs the parent whether the child process succeeded or failed.
47. How can a parent process terminate a child process in Unix-like operating systems?
A parent terminates a child process using the kill() system call.
48. Explain the difference between a process group and a session in Unix-like operating systems.
A process group consists of related processes, while a session contains multiple process groups.
49. Describe how the exec() family of functions is used to replace the current process image with a new one.
The exec() function replaces the current process image with a new executable.
50. What is the purpose of the waitpid() system call in process management? How does it differ from wait()?
The waitpid() system call waits for a specific child process, unlike wait(), which waits for any child.
51. How does process termination occur in Unix-like operating systems?
Process termination occurs when a process finishes execution or is killed.
52. What is the role of the long-term scheduler in the process scheduling hierarchy?
The long-term scheduler controls job admission, influencing system load and multiprogramming level.
53. How does the short-term scheduler differ from the long-term and medium-term schedulers?
The short-term scheduler selects processes frequently, unlike the long-term and medium-term schedulers.
54. Describe a scenario where the medium-term scheduler would be invoked.
The medium-term scheduler suspends processes to optimize memory and CPU usage.

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.

PID	AT	BT	CT	TAT	WT
P1	0	5	5	5	0
P2	1	3	8	7	4
P3	2	6	14	12	6

Average Waiting Time (AWT) AWT=(0+4+6) / 3=10/3=**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.

PID	AT	BT	CT	TAT
P1	0	3	3	3
P3	2	1	4	2
P4	3	4	8	5
P2	1	5	13	12

Average Turnaround Time (ATAT)

ATAT=(3+2+5+12) / 4 = 22/4 = **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.

PID	AT	BT	Priority	CT	TAT	WT
P2	1	4	1	5	4	0
P4	3	2	2	7	4	2
P1	0	6	3	13	13	7
P3	2	7	4	20	18	11

Average Waiting Time (AWT)

AWT=(0+2+7+11) / 4 = 20 / 4 = **5**

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

PID	AT	BT	CT	TAT
P1	0	4	8	8
P2	1	5	14	13
P3	2	2	6	4
P4	3	3	11	8

Average Turnaround Time (ATAT)

ATAT = (8+13+4+8) / 4 = 33 / 4 = **8**.

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?

=> Each process gets a copy of x. Both parent and child increment x by 1 independently.
=> x in Parent = 6
=> x in Child = 6

Submission Guidelines:

- Document each step of your solution and any challenges faced.
- Upload it on your GitHub repository

Additional Tips:

- Experiment with different options and parameters of each command to explore their functionalities.
- This assignment is tailored to align with interview expectations, CCEE standards, and industry demands.
- If you complete this then your preparation will be skyrocketed.