

CDAC MUMBAI

Concepts of Operating System Assignment 2

Part A

What will the following commands do?

1) echo "Hello, World!"

echo → A built-in command in Unix/Linux/macOS that prints text to the terminal.

2) name="Productive"

This command creates a variable named name and assigns it the value "Productive".

```
vaishnavikulkarni — zsh — 119x32
Last login: Sat Mar 1 07:04:17 on console
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % echo 'Hello, World!'

Hello, World!
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % name="Productive"

vaishnavikulkarni@Vaishnavis-MacBook-Air ~ %
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % echo $name

Productive
```

3) touch file.txt

Touch → A command used to create an empty file or update the timestamp of an existing file.

file.txt → The name of the file to be created or updated.

4) ls -a

ls → Lists the contents of a directory.

-a → Shows all files, including hidden files (files starting with . like .bashrc or .git).

```
Productive
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % touch file.txt

vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % ls -l file.txt
-rw-r--r-- 1 vaishnavikulkarni staff 0 Mar 1 11:24 file.txt
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % ls -a
.                  .vscode-react-native  LinuxAssignment      file.txt
..                 .zsh_history           Movies               fruit.t
.CFUserTextEncoding .zsh_sessions         Music                fruit.txt
.DS_Store          Desktop                Pictures             input.txt
.Trash              Documents              Public               numbers.txt
.config             Downloads              data.txt             output.txt
.vscodex            Library                duplicate.txt        unique.txt
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % rm file.txt
```

5) rm file.txt

rm → Removes (deletes) files or directories.

file.txt → The file to be deleted.

6) cp file1.txt file2.txt

cp → Copies files or directories.

file1.txt → The source file (the file to be copied).

file2.txt → The destination file (the new copy).

```

.vscod
Library
duplicate.txt
unique.txt
[vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % rm file.txt

[vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % ls -l file.txt
ls: file.txt: No such file or directory
[vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % cp file1.txt file2.txt

cp: file1.txt: No such file or directory
[vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % touch file1.txt
[vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % touch file2.txt
[vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % cp file1.txt file2.txt

[vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % ls -l file2.txt
-rw-r--r--  1 vaishnavikulkarni  staff  0 Mar  1 11:32 file2.txt
[vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % echo "Hello, World!" > file1.txt

[quote>
[vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % echo 'Hello, World!' > file1.txt
[vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % cp file1.txt file2.txt
[vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % cat file2.txt
Hello, World!
[vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % mv file1.txt /path/to/directory/

```

7) mv file.txt /path/to/directory/

mv → Moves (or renames) files and directories.

file.txt → The file to be moved.

/path/to/directory/ → The destination directory where the file will be moved.

8) chmod 755 script.sh

chmod → Changes file permissions.

755 → Sets specific permissions (rwxr-xr-x).

script.sh → The file whose permissions are being changed.

```

[vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % pwd
/Users/vaishnavikulkarni
[vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % touch script.sh

[vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % chmod 755 script.sh

[vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % ls -l script.sh
-rwxr-xr-x  1 vaishnavikulkarni  staff  0 Mar  1 11:53 script.sh
[vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % █

```

9) grep "pattern" file.txt

grep → Searches for a specific **pattern** in a file.

"pattern" → The text or regular expression to search for.

file.txt → The file in which to search.

```

[vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % nano file.txt
[vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % cat file.txt

Hello, World!
Linux is amazing.
This is a test pattern.
Grep is powerful.

[vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % grep 'pattern' file.txt

This is a test pattern.

```

10) kill PID

kill → Sends a signal to a process to stop it.

PID → The Process ID of the program you want to stop.

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

Creates a directory called mydir.

Changes into that directory.

Creates an empty file called file.txt.

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

Displays the contents of file.txt using cat.

```
mydir — zsh — 132x40
[vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % mkdir mydir && cd mydir && touch file.txt && echo 'Hello, World!' > file.txt && cat file.txt
Hello, World!
vaishnavikulkarni@Vaishnavis-MacBook-Air mydir %
```

12) ls -l | grep ".txt"

ls -l: Lists files in the current directory in long format, showing details like permissions, owner, size, and date modified.

grep ".txt": Filters and displays only the files with a .txt extension.

```
Hello, World!
[vaishnavikulkarni@Vaishnavis-MacBook-Air mydir % ls -l | grep ".txt"]
-rw-r--r-- 1 vaishnavikulkarni staff 14 Mar 1 14:43 file.txt
[vaishnavikulkarni@Vaishnavis-MacBook-Air mydir % touch test1.txt test2.txt]
[vaishnavikulkarni@Vaishnavis-MacBook-Air mydir %]
[vaishnavikulkarni@Vaishnavis-MacBook-Air mydir %]
[vaishnavikulkarni@Vaishnavis-MacBook-Air mydir %]
[vaishnavikulkarni@Vaishnavis-MacBook-Air mydir %]
```

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

cat file1.txt file2.txt: Concatenates the contents of file1.txt and file2.txt.

sort: Sorts the combined content alphabetically.

uniq: Removes duplicate lines, showing only unique lines.

```
[vaishnavikulkarni@Vaishnavis-MacBook-Air mydir % cat file.txt | grep "Hello"]
Hello, World!
[vaishnavikulkarni@Vaishnavis-MacBook-Air mydir % ps aux | grep python]
vaishnavikulkarni 3996  0.0  0.0 410724096 1424 s000  S+   2:52PM   0:00.01 grep python
[vaishnavikulkarni@Vaishnavis-MacBook-Air mydir % ls -l | sort -k 5 -n]
-rw-r--r-- 1 vaishnavikulkarni staff  0 Mar 1 14:47 test1.txt
-rw-r--r-- 1 vaishnavikulkarni staff  0 Mar 1 14:47 test2.txt
total 8
-rw-r--r-- 1 vaishnavikulkarni staff 14 Mar 1 14:43 file.txt
[vaishnavikulkarni@Vaishnavis-MacBook-Air mydir % cat file.txt | wc -l]
1
[vaishnavikulkarni@Vaishnavis-MacBook-Air mydir %]
```

14) ls -l | grep "^d"

ls -l: Lists files and directories in long format.

grep "^d": Filters and displays only directories.

The ^ (caret) in regular expressions means "start of the line".

The d indicates a directory in the permissions column of ls -l.

```

bash: command not found: drwxr-xr-x
vaishnavikul Karni@Vaishnavis-MacBook-Air mydir % cd ~/mydir

vaishnavikul Karni@Vaishnavis-MacBook-Air mydir % ls -l | grep "^d"

vaishnavikul Karni@Vaishnavis-MacBook-Air mydir % mkdir testdir
ls -l | grep "^d"

drwxr-xr-x  2 vaishnavikul Karni  staff   64 Mar  1 14:57 testdir
vaishnavikul Karni@Vaishnavis-MacBook-Air mydir %

```

15) `grep -r "pattern" /path/to/directory/`

`grep`: Searches for text patterns in files.

`-r` (Recursive): Searches recursively through all files and subdirectories.

"pattern": The text or regular expression to search for.

`/path/to/directory/`: The directory to search in.

```

drwxr-xr-x  2 vaishnavikul Karni  staff   64 Mar  1 14:57 testdir
vaishnavikul Karni@Vaishnavis-MacBook-Air mydir % grep -r "Hello" ~/Documents/

/Users/vaishnavikul Karni/Documents//CDAC-OS/Day-2/Linux Folder/abc.txt:Hello
vaishnavikul Karni@Vaishnavis-MacBook-Air mydir %

```

16) `cat file1.txt file2.txt | sort | uniq -d`

`cat file1.txt file2.txt`: Concatenates the contents of both files.

`sort`: Sorts all lines alphabetically (required for `uniq` to work properly).

`uniq -d`: Displays only the duplicate lines, i.e., lines that appear more than once.

```

vaishnavikul Karni@Vaishnavis-MacBook-Air mydir % chmod 644 file.txt

vaishnavikul Karni@Vaishnavis-MacBook-Air mydir % ls -l file.txt

-rw-r--r--  1 vaishnavikul Karni  staff   14 Mar  1 14:43 file.txt
vaishnavikul Karni@Vaishnavis-MacBook-Air mydir %

```

17) `chmod 644 file.txt`

The `chmod` command changes permissions of `file.txt` to 644, which means:

6 = `rw-`: Read and write for the owner.

4 = `r--`: Read-only for the group.

4 = `r--`: Read-only for others.

```

vaishnavikul Karni@Vaishnavis-MacBook-Air mydir % touch file1.txt
vaishnavikul Karni@Vaishnavis-MacBook-Air mydir % touch file2.txt
vaishnavikul Karni@Vaishnavis-MacBook-Air mydir % cat file1.txt file2.txt | sort | uniq -d

vaishnavikul Karni@Vaishnavis-MacBook-Air mydir % cat file1.txt file2.txt | sort | uniq -c | grep -v " 1 "

vaishnavikul Karni@Vaishnavis-MacBook-Air mydir % echo -e "apple\nbanana\ncherry" > file1.txt
echo -e "banana\napple\ndate" > file2.txt

vaishnavikul Karni@Vaishnavis-MacBook-Air mydir % cat file1.txt file2.txt | sort | uniq -d

apple
banana
vaishnavikul Karni@Vaishnavis-MacBook-Air mydir % cat file1.txt
cat file2.txt

apple
banana
cherry
banana
apple
date
vaishnavikul Karni@Vaishnavis-MacBook-Air mydir %

```

18) `cp -r source_directory destination_directory`

`cp`: Copies files and directories.

-r (Recursive): Copies the directory recursively, including:
All files
Subdirectories
Their contents
source_directory: The directory you want to copy.
destination_directory: The target location where the directory and its contents will be copied.

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

find: Searches for files and directories in a specified path.

/path/to/search: The directory where the search starts (e.g., /home/user, . for the current directory, or / for the entire filesystem).

-name "*.txt": Finds files with names matching the pattern "*.txt".

*.txt matches all files with a .txt extension.

The pattern is case-sensitive by default.

```
find: /path/to/search: No such file or directory
vaishnavikulkarni@Vaishnavis-MacBook-Air mydir % find . -name "*.txt"

./file2.txt
./file.txt
./file1.txt
./test1.txt
./test2.txt
vaishnavikulkarni@Vaishnavis-MacBook-Air mydir %
```

20) chmod u+x file.txt

chmod: Changes file permissions.

u+x: Adds **execute** (x) permission for the **user** (u, i.e., the file owner).

file.txt: The target file.

```
vaishnavikulkarni@Vaishnavis-MacBook-Air mydir % chmod u+x file.txt

vaishnavikulkarni@Vaishnavis-MacBook-Air mydir % ls -l file.txt

-rwxr--r-- 1 vaishnavikulkarni staff 14 Mar 1 14:43 file.txt
vaishnavikulkarni@Vaishnavis-MacBook-Air mydir %
```

21) echo \$PATH

echo: Displays text or variables in the terminal.

\$PATH: An environment variable that stores a colon-separated list of directories where the shell looks for executable files.

```
-rwxr--r-- 1 vaishnavikulkarni staff 14 Mar 1 14:43 file.txt
vaishnavikulkarni@Vaishnavis-MacBook-Air mydir % echo $PATH
/usr/local/bin:/System/Cryptexes/App/usr/bin:/usr/bin:/bin:/usr/sbin:/sbin:/var/run/com.apple.security.cryptexd/codex.system/bootstrap/usr/local/bin:/var/run/com.apple.security.cryptexd/codex.system/bootstrap/usr/bin:/var/run/com.apple.security.cryptexd/codex.system/bootstrap/usr/appleinternal/bin
vaishnavikulkarni@Vaishnavis-MacBook-Air mydir %
```

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:

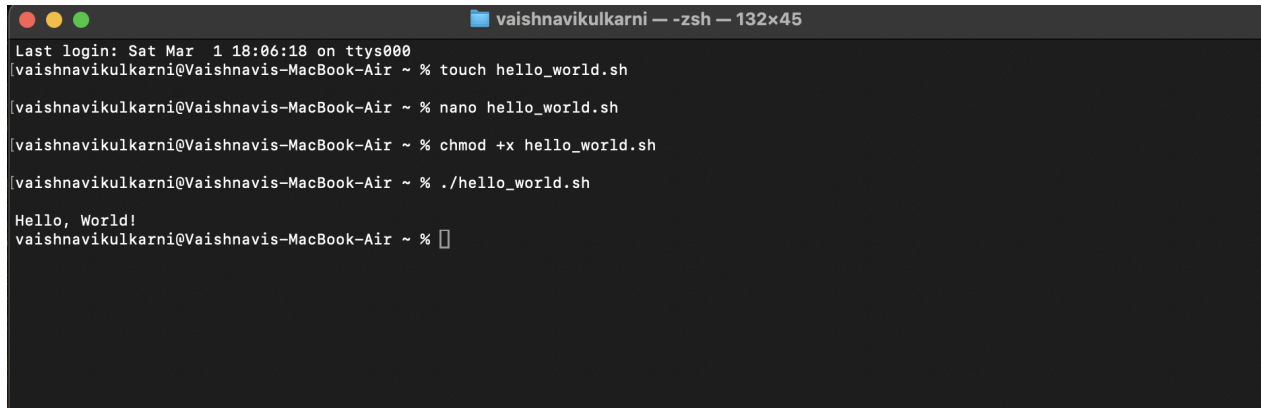
1. chmodx is used to change file permissions. **Incorrect**
chmodx is used to change file permissions.
Correct Command: chmod
Example: chmod 755 file.txt
Description: Changes file permissions using numeric or symbolic modes.
2. cpy is used to copy files and directories. **Incorrect**
cp file1 file2 copies a file.
cp -r dir1 dir2 copies directories recursively.
3. mkfile is used to create a new file. **Incorrect**
The correct command to create a new file in Linux is usually touch, not mkfile.
Example: touch filename.txt creates an empty file named filename.txt The mkfile command exists in some Unix-like systems (e.g., macOS) to create a file of a specific size:
Example: mkfile 1m file1 creates a 1 MB file named file1. However, mkfile is not a standard Linux command.
4. catx is used to concatenate files. **Incorrect**
The correct command to concatenate files in Linux is cat, not catx.
Example: cat file1 file2 > combined.txt combines the contents of file1 and file2 into combined.txt. The cat command is also commonly used to display the contents of a file:
Example: cat filename.txt displays the content of filename.txt.
5. rn is used to rename files. :- **Incorrect**
rn is used to rename files.
Correct Command: mv

Example: mv oldname.txt newname.txt

Description: Moves or renames files and directories.

Part C

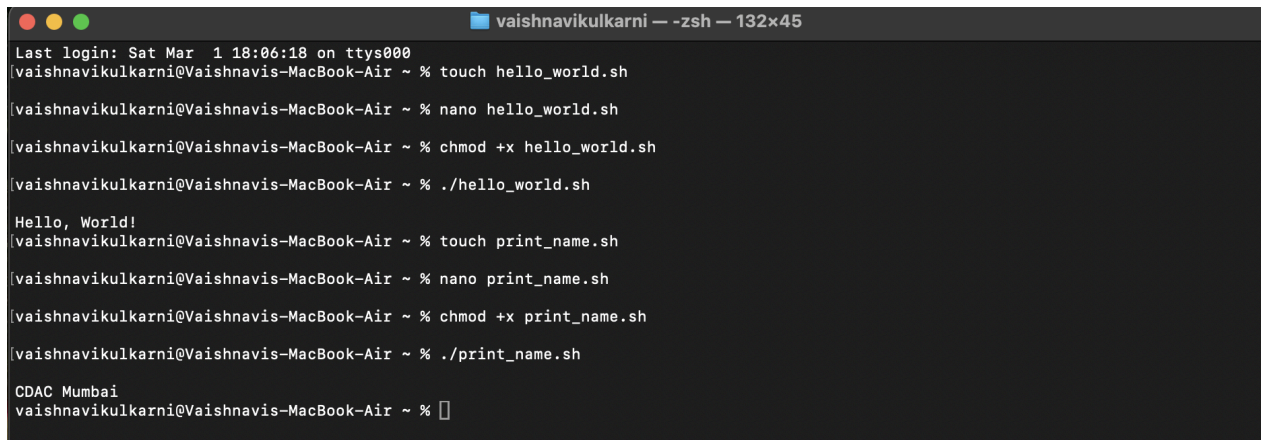
Question 1: Write a shell script that prints "Hello, World!" to the terminal.



```
vaishnavikulkarni — -zsh — 132x45
Last login: Sat Mar 1 18:06:18 on ttys000
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % touch hello_world.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % nano hello_world.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % chmod +x hello_world.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % ./hello_world.sh

Hello, World!
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ %
```

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



```
vaishnavikulkarni — -zsh — 132x45
Last login: Sat Mar 1 18:06:18 on ttys000
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % touch hello_world.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % nano hello_world.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % chmod +x hello_world.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % ./hello_world.sh

Hello, World!
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % touch print_name.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % nano print_name.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % chmod +x print_name.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % ./print_name.sh

CDAC Mumbai
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ %
```

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

```
CDAC Mumbai
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % touch print_number.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % nano print_number.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % chmod +x print_number.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % ./print_number.sh

Enter a number:
23
You entered: 23
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ %
```

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

```
You entered: 23
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % touch add_numbers.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % nano add_numbers.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % chmod +x add_numbers.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % ./add_numbers.sh

The sum of 5 and 3 is: 8
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ %
```

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

```
The sum of 5 and 3 is: 8
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % touch even_odd.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % nano even_odd.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % chmod +x even_odd.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % ./even_odd.sh

Enter a number:
5
Odd
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ %
```

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

```
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % nano print_numbers.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % chmod +x print_numbers.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % ./print_numbers.sh

1
2
3
4
5
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ %
```


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

```
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % touch while_loop.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % nano while_loop.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % chmod +x while_loop.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % ./while_loop.sh
1
2
3
4
5
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ %
```

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".

```
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % touch check_file.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % nano check_file.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % chmod +x check_file.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % ./check_file.sh
File exists
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ %
```

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.

```
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % touch check_number.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % nano check_number.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % chmod +x check_number.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % ./check_number.sh
Enter a number:
45
The number 45 is greater than 10.
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ %
```

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.

```
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % chmod +x multiplication_table.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % ./multiplication_table.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
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ %
```

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.

```
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % touch square_numbers.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % nano square_numbers.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % chmod +x square_numbers.sh
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ % ./square_numbers.sh

Enter a number (negative to quit):
5
Square of 5 is 25
Enter a number (negative to quit):
6
Square of 6 is 36
Enter a number (negative to quit):
-2
Negative number entered. Exiting...
vaishnavikulkarni@Vaishnavis-MacBook-Air ~ %
```

Part D

Common Interview Questions (Must know)

1. What is an operating system, and what are its primary functions?
2. Explain the difference between process and thread.
3. What is virtual memory, and how does it work?
4. Describe the difference between multiprogramming, multitasking, and multiprocessing.
5. What is a file system, and what are its components?
6. What is a deadlock, and how can it be prevented?
7. Explain the difference between a kernel and a shell.
8. What is CPU scheduling, and why is it important?
9. How does a system call work?
10. What is the purpose of device drivers in an operating system?
11. Explain the role of the page table in virtual memory management.
12. What is thrashing, and how can it be avoided?
13. Describe the concept of a semaphore and its use in synchronization.
14. How does an operating system handle process synchronization?
15. What is the purpose of an interrupt in operating systems?
16. Explain the concept of a file descriptor.
17. How does a system recover from a system crash?
18. Describe the difference between a monolithic kernel and a microkernel.
19. What is the difference between internal and external fragmentation?
20. How does an operating system manage I/O operations?
21. Explain the difference between preemptive and non-preemptive scheduling.
22. What is round-robin scheduling, and how does it work?
23. Describe the priority scheduling algorithm. How is priority assigned to processes?
24. What is the shortest job next (SJN) scheduling algorithm, and when is it used?
25. Explain the concept of multilevel queue scheduling.

26. What is a process control block (PCB), and what information does it contain?
27. Describe the process state diagram and the transitions between different process states.
28. How does a process communicate with another process in an operating system?
29. What is process synchronization, and why is it important?
30. Explain the concept of a zombie process and how it is created.
31. Describe the difference between internal fragmentation and external fragmentation.
32. What is demand paging, and how does it improve memory management efficiency?
33. Explain the role of the page table in virtual memory management.
34. How does a memory management unit (MMU) work?
35. What is thrashing, and how can it be avoided in virtual memory systems?
36. What is a system call, and how does it facilitate communication between user programs and the operating system?
37. Describe the difference between a monolithic kernel and a microkernel.
38. How does an operating system handle I/O operations?
39. Explain the concept of a race condition and how it can be prevented.
40. Describe the role of device drivers in an operating system.
41. What is a zombie process, and how does it occur? How can a zombie process be prevented?
42. Explain the concept of an orphan process. How does an operating system handle orphan processes?
43. What is the relationship between a parent process and a child process in the context of process management?
44. How does the fork() system call work in creating a new process in Unix-like operating systems?
45. Describe how a parent process can wait for a child process to finish execution.
46. What is the significance of the exit status of a child process in the wait() system call?
47. How can a parent process terminate a child process in Unix-like operating systems?
48. Explain the difference between a process group and a session in Unix-like operating systems.
49. Describe how the exec() family of functions is used to replace the current process image with a new one.
50. What is the purpose of the waitpid() system call in process management? How does it differ from wait()?
51. How does process termination occur in Unix-like operating systems?
52. What is the role of the long-term scheduler in the process scheduling hierarchy? How does it influence the degree of multiprogramming in an operating system?
53. How does the short-term scheduler differ from the long-term and medium-term schedulers in terms of frequency of execution and the scope of its decisions?
54. Describe a scenario where the medium-term scheduler would be invoked and explain how it helps manage system resources more efficiently.

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.

Q.1. Algorithm used : FCFS

Ans:

process	Arrival time	Burst time	Waiting time
P1	0	5	0
P2	1	3	4
P3	2	6	6

Grant chart:

P1	P2	P3	
0	5	8	14

Avg. waiting time = $\frac{(0+4+6)}{3} = 10/3$

Avg. waiting time = 3.333 units.

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.

Q.2. Algorithm used: SJF (non-pre-emptive).

Ans:

process	Arrival time	B.T	Waiting time	T.A.T.
P1	0	3	0	3
P2	1	5	7	12
P3	2	1	1	2
P4	3	4	1	5

Gantt chart:

```

  0   3   4   8   13
  | P1 | P3 | P4 | P2 |
  
```

Avg. turn around time = $\frac{(3 + 12 + 2 + 5)}{4} = \frac{22}{4} = 5.5$ units

Avg. turnaround time = 5.5 units.

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.

Q.3. Algorithm used: Priority scheduling (non-pre-emptive)

Ans:

Process	A.T	B.T	Completion time	TAT	Waiting time
P1	0	6	6	6	0
P2	1	4	10	9	5
P3	2	7	19	17	10
P4	3	2	12	9	7

Avg. waiting time = $\frac{(0+5+7+10)}{4} = \frac{22}{4} = 5.5$ units.

for pre-emptive Priority scheduling:-

Process	Arrival time	B.T	Completion time	TAT	W.T
P1	0	6	12	12	6
P2	1	4	5	4	0
P3	2	7	19	17	10
P4	3	2	7	4	2

Grant chart: P1 | P2 | P4 | P3 | P3

Avg. waiting time = $\frac{(6+0+10+2)}{4} = \frac{18}{4} = 4.5$ units.

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.

Q.4: Algorithm used: Round Robin Scheduling

Ans: Quantum = 2 units.

Process	AT	BT	TAT	WT
P1	0	4	10	4
P2	1	5	13	8
P3	2	2	4	2
P4	3	3	10	7

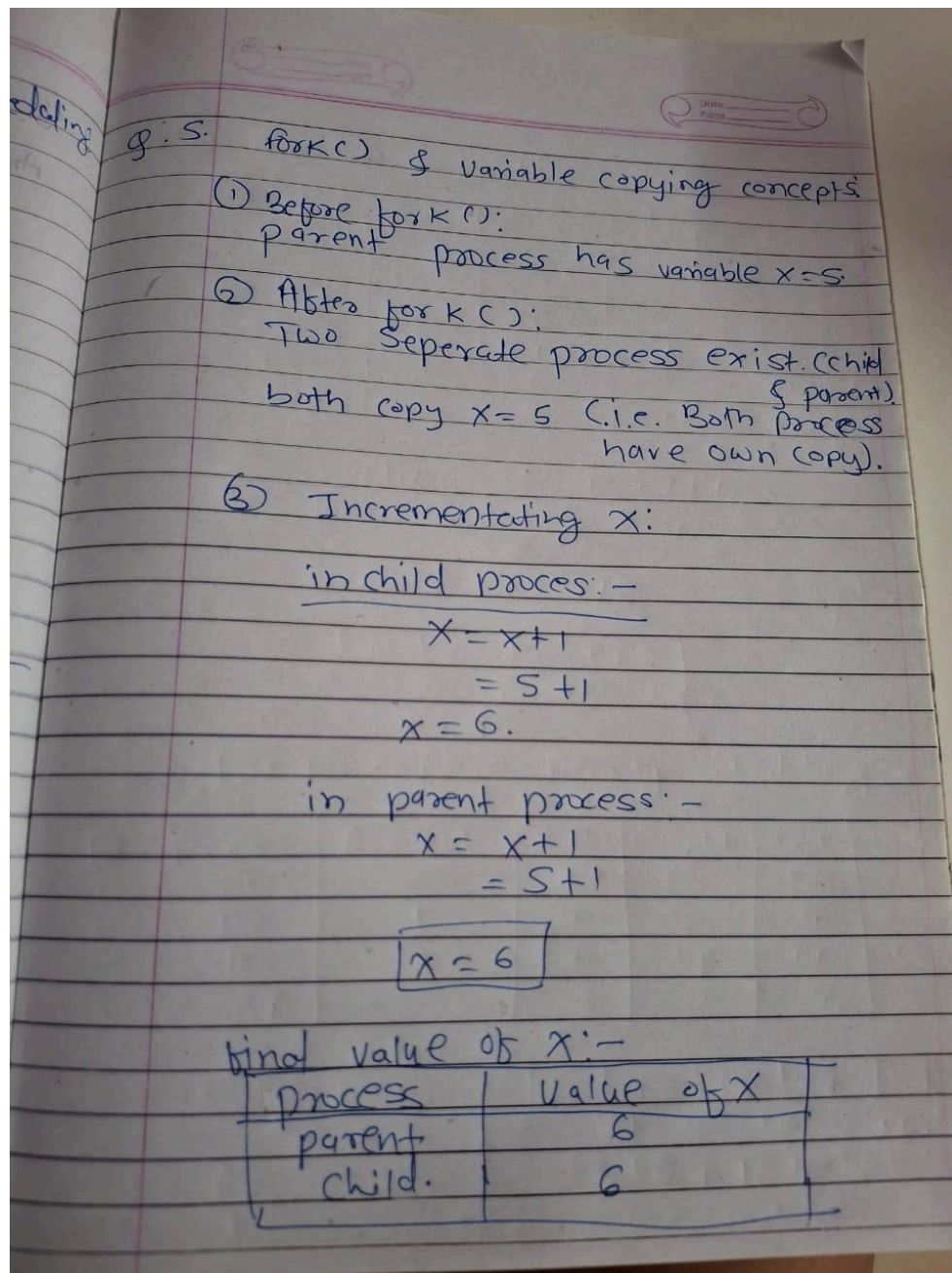
Gantt chart:

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

Avg. Turnaround Time = $\frac{10 + 13 + 4 + 10}{4} = 9.25$

Avg. TAT = 9.25 units.

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?



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.