Assignment 2

Part A

echo "Hello, World!"
→ Prints "Hello, World!" to the terminal.

name="Productive"
→ Assigns the value "Productive" to the variable 'name'.

touch file.txt
→ Creates an empty file named 'file.txt' if it doesn’t exist.

ls -a
→ Lists all files, including hidden ones, in the current directory.

rm file.txt
→ Deletes the file 'file.txt'.

cp file1.txt file2.txt
→ Copies 'file1.txt' to 'file2.txt'.

mv file.txt /path/to/directory/
→ Moves 'file.txt' to the specified directory.

chmod 755 script.sh
→ Grants execute permission to all and write permission only to the owner.

grep "pattern" file.txt
→ Searches for "pattern" inside 'file.txt' and displays matching lines.

kill PID
→ Terminates the process with the given Process ID (PID).

mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt
→ Creates 'mydir', moves into it, creates 'file.txt', writes "Hello, World!" to it, and displays its content.

ls -l | grep ".txt"
→ Lists detailed info of files ending with '.txt'.

cat file1.txt file2.txt | sort | uniq
→ Merges, sorts, and removes duplicate lines from two files.

ls -l | grep "^d"
→ Lists only directories in the current directory.

grep -r "pattern" /path/to/directory/
→ Recursively searches for "pattern" in all files in the directory.

cat file1.txt file2.txt | sort | uniq –d
→ Displays duplicate lines in both files.

chmod 644 file.txt
→ Grants read and write permission to the owner, read-only to others.

cp -r source_directory destination_directory
→ Recursively copies a directory to another location.

find /path/to/search -name "*.txt"
→ Searches for all '.txt' files in the given path.

chmod u+x file.txt
→ Grants execute permission to the file owner.

echo $PATH
→ Displays the system's executable search paths.

Part B

True or False

1. True → ls is used to list files and directories in a directory.
2. True → mv is used to move files and directories.
3. False → cd is used to change directories, not copy files and directories.
4. True → pwd stands for "print working directory" and displays the current directory.
5. True → grep is used to search for patterns in files.
6. True → chmod 755 file.txt gives read, write, and execute permissions to the owner, and read and execute permissions to group and others.
7. True → mkdir -p directory1/directory2 creates nested directories, creating directory2 inside directory1 if directory1 does not exist.
8. True → rm -rf file.txt deletes a file forcefully without confirmation.

Identify the correct command:

1. Incorrect → chmodx is not a valid command; the correct command is chmod.
2. Incorrect → cpy is not a valid command; the correct command is cp.
3. Incorrect → mkfile is not a standard Linux command; use touch to create a new file.
4. Incorrect → catx is not a valid command; the correct command is cat.
5. Incorrect → rn is not a valid command; use mv to rename files.

Part C

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

→ `#!/bin/bash`
  `echo "Hello, World!"`

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

→ `#!/bin/bash`
  `name="CDAC Mumbai"`
  `echo "The value of name is: $name"`

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

→ `#!/bin/bash`
  `read -p "Enter a number: " num`
  `echo "You entered: $num"`

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

→ `#!/bin/bash`
  `num1=5`
  `num2=3`
  `sum=$((num1 + num2))`
  `echo "The sum is: $sum"`

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

→ `#!/bin/bash`
  `read -p "Enter a number: " num`
  `if (( num % 2 == 0 )); then`
  `    echo "Even"`
  `else`
  `    echo "Odd"`
  `fi`

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

→ `#!/bin/bash`

`for i in {1..5}; do`
`    echo $i`
`done`

Question 7: Write a shell script that uses a while loop to print numbers from 1 to 5.
→ `#!/bin/bash`
`i=1`
`while [ $i -le 5 ]; do`
`    echo $i`
`    ((i++))`
`done`

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".
→ `#!/bin/bash`
`if [ -f "file.txt" ]; then`
`    echo "File exists"`
`else`
`    echo "File does not exist"`
`fi`

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.
→ `#!/bin/bash`
`read -p "Enter a number: " num`
`if (( num > 10 )); then`
`    echo "The number is greater than 10"`
`else`
`    echo "The number is 10 or less"`
`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.

→ `#!/bin/bash`
  `for i in {1..5}; do`
  `    for j in {1..5}; do`
  `        printf "%4d" $(( i * j ))`
  `    done`
  `    echo`
  `done`

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.

→ `#!/bin/bash`
  `while true; do`
  `    read -p "Enter a number: " num`
  `    if (( num < 0 )); then`
  `        echo "Negative number entered. Exiting..."`
  `        break`
  `    fi`
  `    echo "Square: $(( num * num ))"`
  `done`

Part E

QUE1->

Step 1: Calculate Completion Time (CT)

In First-Come, First-Served (FCFS) scheduling, processes are executed in the order of their arrival time.

- **P1 arrives at 0, starts immediately, and finishes at 0+5=50 + 5 = 50+5=5.**
- **P2 arrives at 1, starts after P1 finishes (at 5), and finishes at 5+3=85 + 3 = 85+3=8.**
- **P3 arrives at 2, starts after P2 finishes (at 8), and finishes at 8+6=148 + 6 = 148+6=14.**

Step 2: Calculate Waiting Time (WT)

WT=CT−AT−BTWT = CT - AT - BTWT=CT−AT−BT

Step 3: Calculate Average Waiting Time

Average WT=∑WTTotal Processes\text{Average WT} = \frac{\sum WT}{\text{Total Processes}}Average WT=Total Processes∑WT =0+4+63=103=3.33 ms= \frac{0 + 4 + 6}{3} = \frac{10}{3} = 3.33 \text{ ms}=30+4+6=310=3.33 ms

QUE2:

- At time **0**, **P1** is the only available process, so it executes first.
- At time **3**, **P2, P3, P4** are available. The shortest job is **P3** (1 unit).
- At time **4**, **P2 and P4** are left. The shortest job is **P4** (4 units).
- At time **8**, only **P2** is left, so it executes last.

Execution Order:
P1 → P3 → P4 → P2

Calculate Turnaround Time (TAT)
T
A
T
=
C
T
−
A
T
TAT=CT−AT

Calculate Average Turnaround Time
Average TAT=∑TATTotal Processes\text{Average TAT} = \frac{\sum TAT}{\text{Total Processes}}Average TAT=Total Processes∑TAT =3+2+5+124=224=5.5 ms= \frac{3 + 2 + 5 + 12}{4} = \frac{22}{4} = 5.5 \text{ ms}=43+2+5+12=422=5.5 m

Final Answers:

1. **FCFS Average Waiting Time = 3.33 ms**

2. **SJF Average Turnaround Time = 5.5 ms**