Assignment No. 2
Part A

1. echo "Hello,World!" :- Prints "Hello,World!" to the console.
2. name="Productive" :- Assigns the value "Productive" to the variable name.
3. touch file.txt :- Creates an empty file name "file.txt".
4. ls -a :- Lists all files, including hidden ones, in the current directory.
5. rm file.txt :- Removes (deletes) the file "file.txt".
6. cp file1.txt file2.txt :- Copies "file1.txt" to "file2.txt".
7. mv file.txt /path/to/directory/ :- Moves "file.txt" to the specified directory.
8. chmod 755 script.sh :- Changes the permissions of "script.sh" to allow execution.
9. grep "pattern" file.txt :- Searches for "pattern" in the content of "file.txt".
10. kill PID :- Sends a signal to terminate the process with the specified process ID (PID).
11. mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt
:- Creates a directory, enters it, creates a file, writes "Hello, World!" to the file, and then displays the file content.
12. ls -l | grep ".txt" :- Lists only .txt file in the current directory.
13. cat file1.txt file2.txt | sort | uniq :- Concatenates the content of "file1.txt" and "file2.txt," sorts it, and removes duplicate lines.
14. ls -l | grep "^d" :- Lists only directories in the current directory.
15. grep -r "pattern" /path/to/directory/ :- Recursively searches for "pattern" in files within the specified directory.
16. cat file1.txt file2.txt | sort | uniq –d :- Displays only the duplicate lines common to both files.
17. chmod 644 file.txt :- Sets read and write permissions for the owner and read-only permissions for others on "file.txt".
18. cp -r source_directory destination_directory :- Copies the contents of the source directory and its subdirectories to the specified destination directory.
The -r flag stands for "recursive," allowing the copying of directories and their contents.
19. find /path/to/search -name "*.txt" :- Searches for files with the extension ".txt" within the specified path (and its subdirectories).
This command uses the find tool to locate files based on criteria, in this case, files with names ending in ".txt".
20. chmod u+x file.txt :- Grants execute permission to the file owner.
21. echo $PATH :- Prints the 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. - False
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: The correct command is chmod.
2. cpy is used to copy files and directories. - Incorrect: The correct command is cp.
3. mkfile is used to create a new file. - Correct
4. catx is used to concatenate files. - Incorrect: The correct command is cat.
5. rn is used to rename files. - Incorrect: The correct command is mv for renaming 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 the variable 'name' is: $name"
```

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

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

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
result=$((num1 + num2))
echo "The result of addition is: $result"
```

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

```
#!/bin/bash
echo "Enter a number: "
read number
if [ $((number % 2)) -eq 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
counter=1
while [ $counter -le 5 ]
do
    echo $counter
    ((counter++))
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 [ -e "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
echo "Enter a number: "
read number
if [ $number -gt 10 ]; then
    echo "The number is greater than 10"
else
    echo "The 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.

```
#!/bin/bash
for i in {1..5}
do
    for j in {1..5}
    do
        result=$((i * j))
        echo -n "$result "
    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

echo "Enter positive numbers (enter a negative number to exit):"

while true; do
    echo -n "Enter a number: "
    read number

    if [ $number -lt 0 ]; then
        echo "Exiting the loop."
        break
    elif [ $number -ge 0 ]; then
        square=$((number * number))
        echo "Square of $number is: $square"
    else
        echo "Invalid input. Please enter a number."
    fi
done
```