

Concepts of Operating System

Assignment 2

Part A

Q.1] What will the following commands do?

-->>

1. `echo "Hello, World!"`

--> This command will print Hello, World!"

2. `name="Productive"`

-->> The variable "name" would get the value "Productive", and

we can refer it with `$name _` {so we will get the value of name i.e

"Productive" }

3. `touch file.txt`

-->> This command will create an empty file with name `file.txt`

4. `ls -a`

-->> This command will show the list of all files & directories in the current Directory & the hidden files as well.

5. `rm file.txt`

-->> This command helps in deleting "`file.txt`" file & we must be carefull as this is irreversible.

6. `cp file1.txt file2.txt`

-->> This command copies the content of `file1.txt` to `file2.txt`

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

-->> This command will move the `file.txt` to the directory.

8. `chmod 755 script.sh`

-->> This command will change the permission of file "script.sh" to 755 ____{ 755 is -> Owner, Group & Others can read, write & execute the permissions. }

9. `grep "pattern" file.txt`

-->> This command will search for string "pattern" in file.txt & will print the lines that contain this string on terminal.

10. `kill PID`

-->> This command sends a signal to process with ID PID, which tells it to stop running. PID is the number assigned to the process we want to target. Like, `kill 1234` would stop the process with ID 1234.

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

-->> `mkdir` will create a new directory named _"mydir". &&
`cd mydir` command will allow changes into mydir directory. &&

touch file.txt will create a empty file named file.txt .

&&

echo "Hello,World!" > file.txt command will write a text

Hello,World! into the file.txt .

&&

cat file.txt command will Display the content of the file .

{&& operator makes the multiple commands work together.}

12. `ls -l | grep ".txt"`

-->> `ls -l | grep ".txt"` this command will display the deatiled list of files specifically the files of .txt will be displayed.

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

-->> This command concatenates the content of file1.txt & file2.txt, then it sorts the combined output & displays it without duplicate lines.

14. `ls -l | grep "^d"`

-->> This command lists directories only in the current directory by filtering `ls -l` output and where the first character "d" (indicates a directory).

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

-->> This command recursively searches for the string "pattern" within all files in the specified directory.

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

-->> This command concatenates `file1.txt` & `file2.txt` then sorts the combined output, & displays only duplicate lines.

17. `chmod 644 file.txt`

-->> This command sets the permissions of `file.txt` to 644, this means the owner can read & write the file, while the group & others can only read it.

18. `cp -r source_directory destination_directory`

-->> This command copies the source directory & all its contents

recursively to destination directory.

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

-->> This command searches for files with a .txt extension under the specified path & its subdirectories.

20. `chmod u+x file.txt`

-->> This command adds execute permissions for the file owner (the user) on file.txt

21. `echo $PATH`

-->> This command displays the current value of the PATH environment variable, which shows the directories & where the system looks for executable files.

Part B

Q.1) 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, *cd* is used to change directories, and not used to copy files & directories. The "*cp*" command is used for copying.**

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.**

Q.2) Identify the Incorrect Commands:

1. chmodx is used to change file permissions.

-->> "chmod" is used to change the file permissions, {chmodx is incorrect.}

2. cpy is used to copy files and directories.

-->> "cp" is the command used to copy files & directories, {cpy is incorrect.}

3. mkfile is used to create a new file.

-->> "touch filename" is the standard command/way to create a new file, and also it can be created by "echo "" > filename", {mkfile is incorrect.}

4. catx is used to concatenate files.

-->> cat is used to concatenate, {catx is incorrect.}

5. rn is used to rename files.

-->> mv is used to rename the files,{rn is incorrect.}

Part C

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

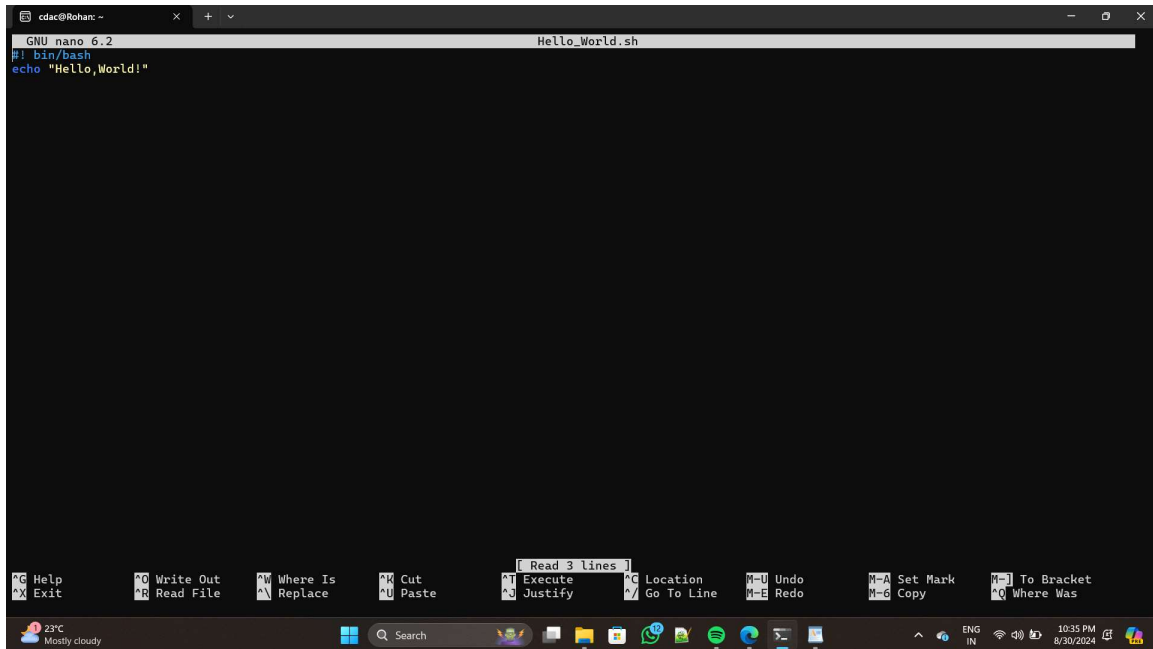
-->>

```
nano Hello_World.sh
```

```
#!/bin/bash
```

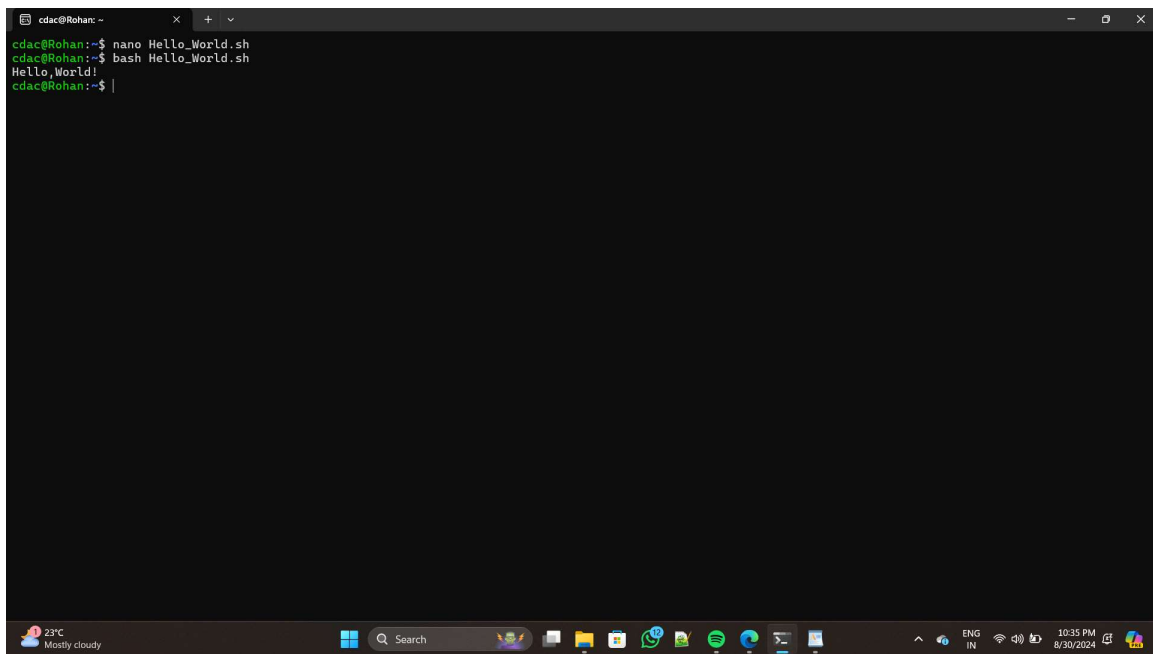
```
echo "Hello_World!"
```

```
bash Hello_World.sh
```



```
GNU nano 6.2 Hello_World.sh
#!/bin/bash
echo "Hello,World!"
```

The screenshot shows a terminal window with the nano 6.2 editor open. The file being edited is Hello_World.sh. The content of the file is: `#!/bin/bash` and `echo "Hello,World!"`. The terminal window has a dark background and a light-colored text. The nano editor's status bar at the bottom shows various keyboard shortcuts like `Ctrl+H` for Help, `Ctrl+X` for Exit, `Ctrl+W` for Write Out, etc. The system tray at the bottom of the window shows the date and time as 10:35 PM on 8/30/2024.



```
cdac@Rohan:~$ nano Hello_World.sh
cdac@Rohan:~$ bash Hello_World.sh
Hello, World!
cdac@Rohan:~$
```

The screenshot shows a terminal window where the user has edited the Hello_World.sh file using nano. The user then runs the command `bash Hello_World.sh`, which outputs `Hello, World!`. The terminal window shows the prompt `cdac@Rohan:~$` before and after the command execution. The system tray at the bottom of the window shows the date and time as 10:35 PM on 8/30/2024.

Question 2: Declare a variable named "name" and assign the value

"CDAC Mumbai" to it. Print the value of the variable.

-->>

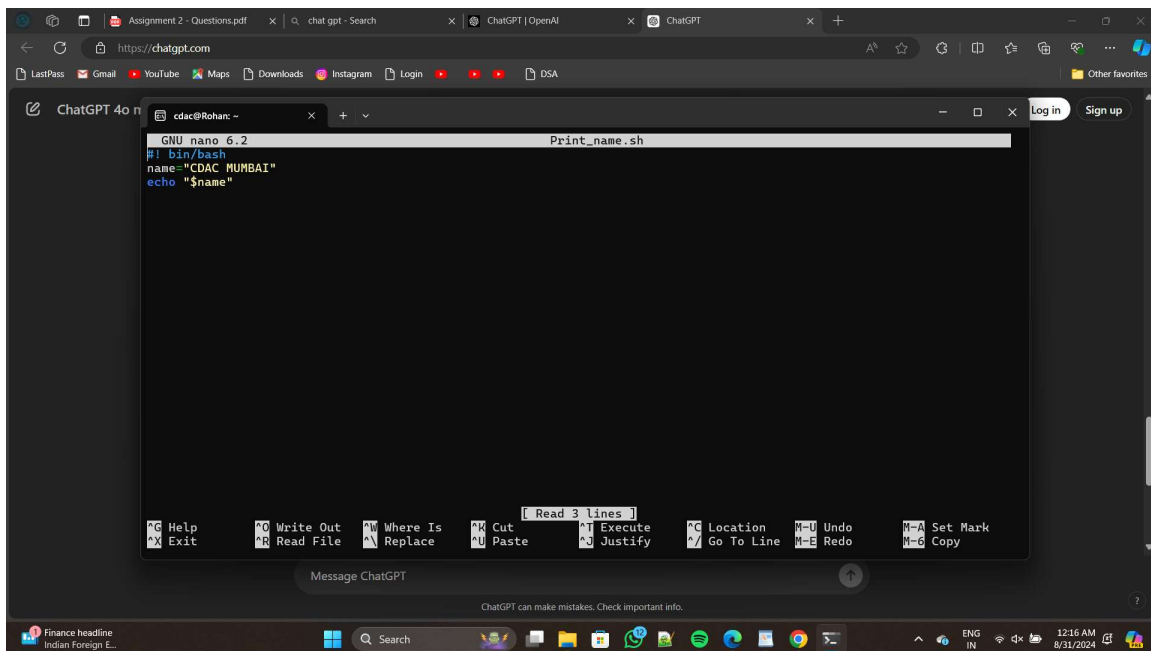
nano Print_name.sh

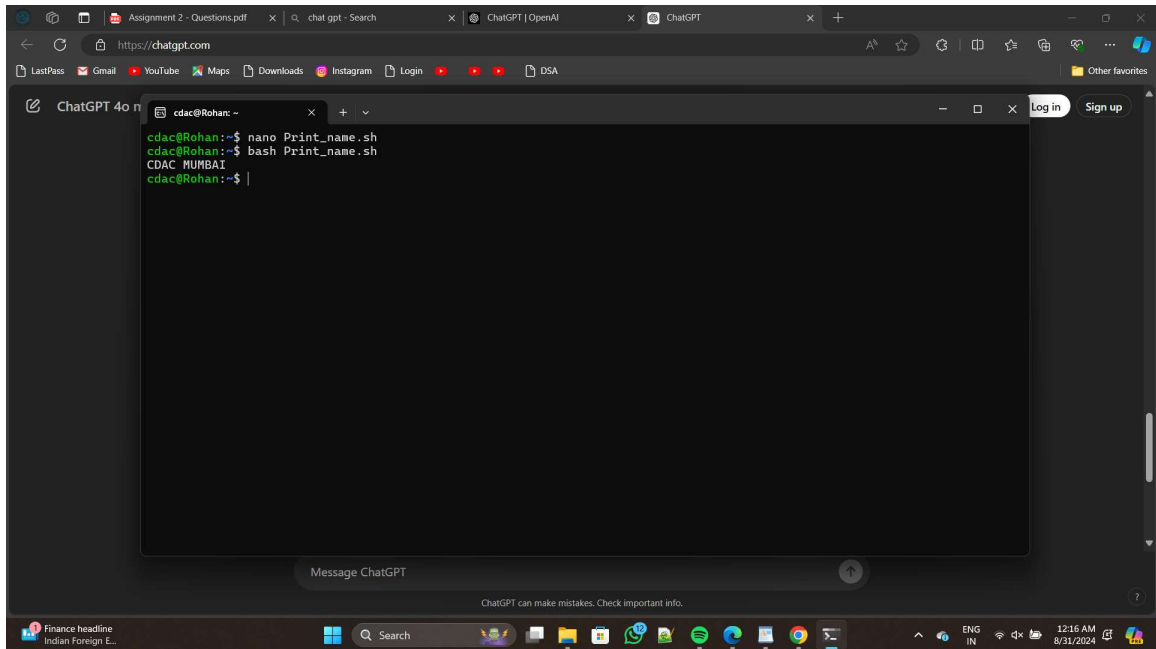
```
#!/bin/bash
```

```
name="CDAC MUMBAI"
```

```
echo "$name"
```

bash Print_name.sh





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

-->>

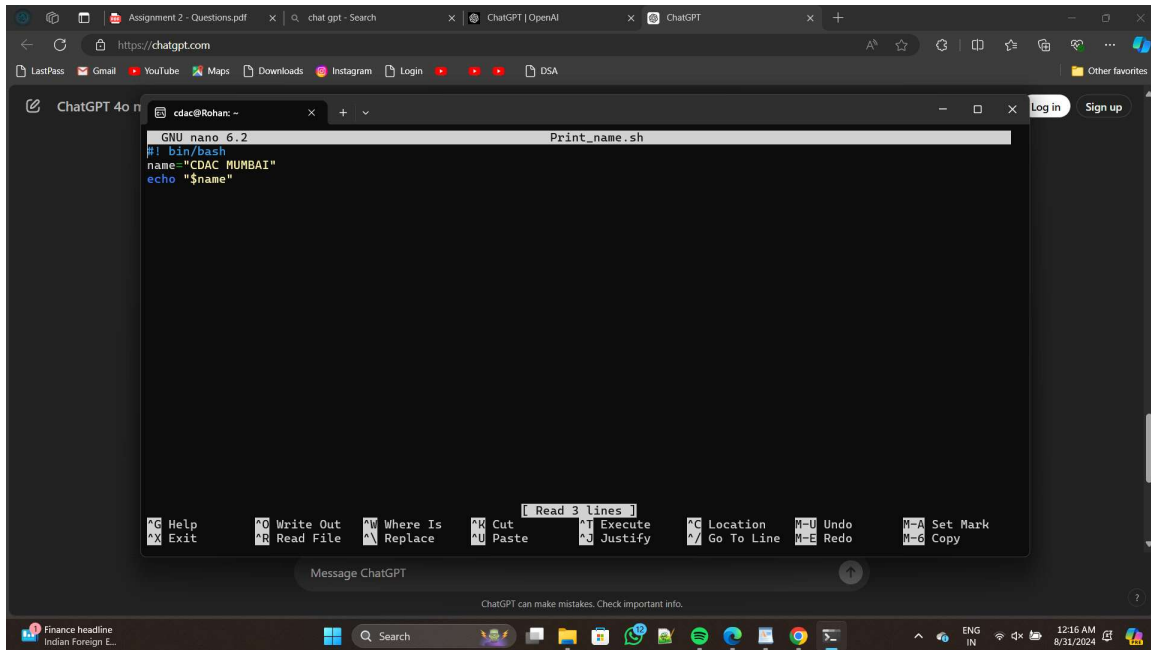
nano print_number.sh

#!/bin/bash

echo "Enter you number :"

read number

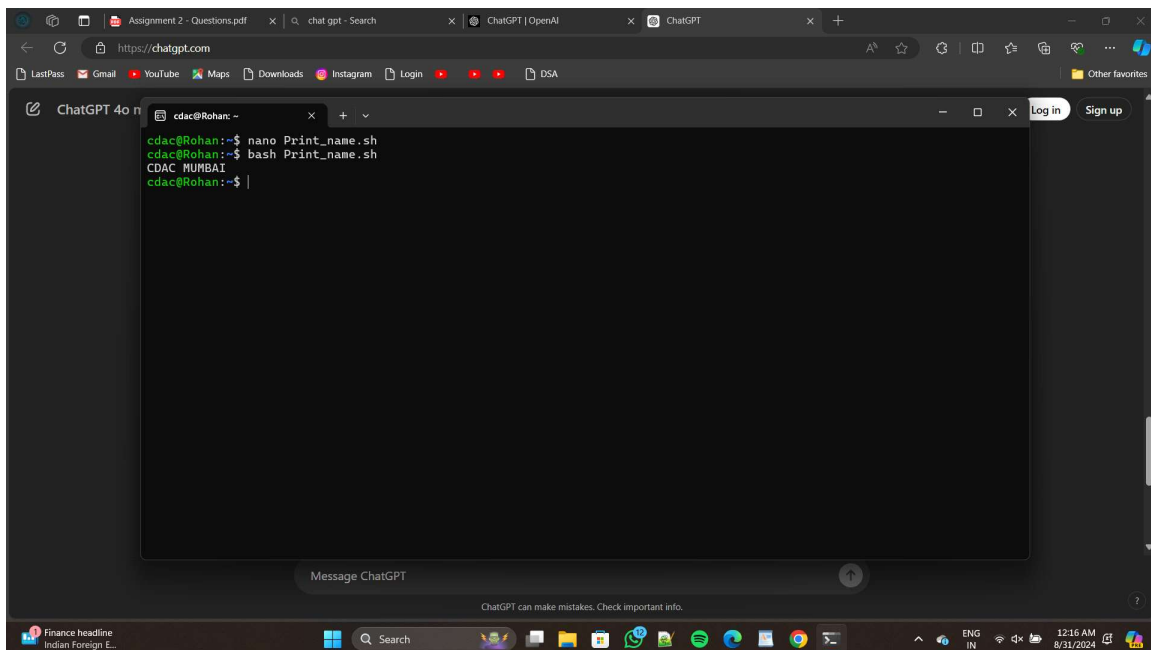
echo "You entered: \$number"



The screenshot shows a terminal window titled "GNU nano 6.2 Print_name.sh". The terminal content is as follows:

```
GNU nano 6.2 Print_name.sh
#!/bin/bash
name="CDAC MUMBAI"
echo "$name"
```

The terminal window is part of a web application interface, with a "Message ChatGPT" button at the bottom. The browser's address bar shows "https://chatgpt.com".



The screenshot shows the same terminal window after the script has been executed. The terminal content is as follows:

```
cdac@Rohan:~$ nano Print_name.sh
cdac@Rohan:~$ bash Print_name.sh
CDAC MUMBAI
cdac@Rohan:~$
```

The terminal window is part of a web application interface, with a "Message ChatGPT" button at the bottom. The browser's address bar shows "https://chatgpt.com".

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

-->>

```
nano add_numbers.sh
```

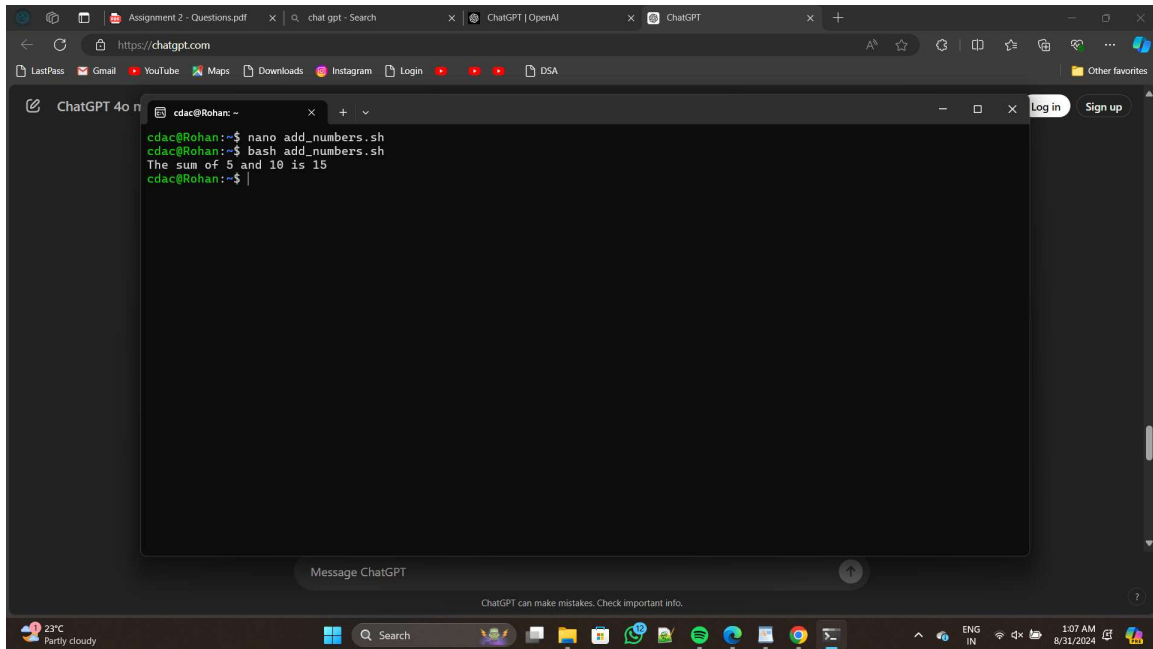
```
#!/bin/bash
```

```
num1=5
```

```
num2=10
```

```
sum=$((num1+num2))
```

```
echo "The sum of $num1 and $num2 is $sum"
```



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

```
-->> nano p5
```

```
#!/bin/bash
```

```
read -p "Enter a number: " number
```



```
if [  $$(number \% 2)$  -eq 0 ]; then
```

```
    echo "Even"
```

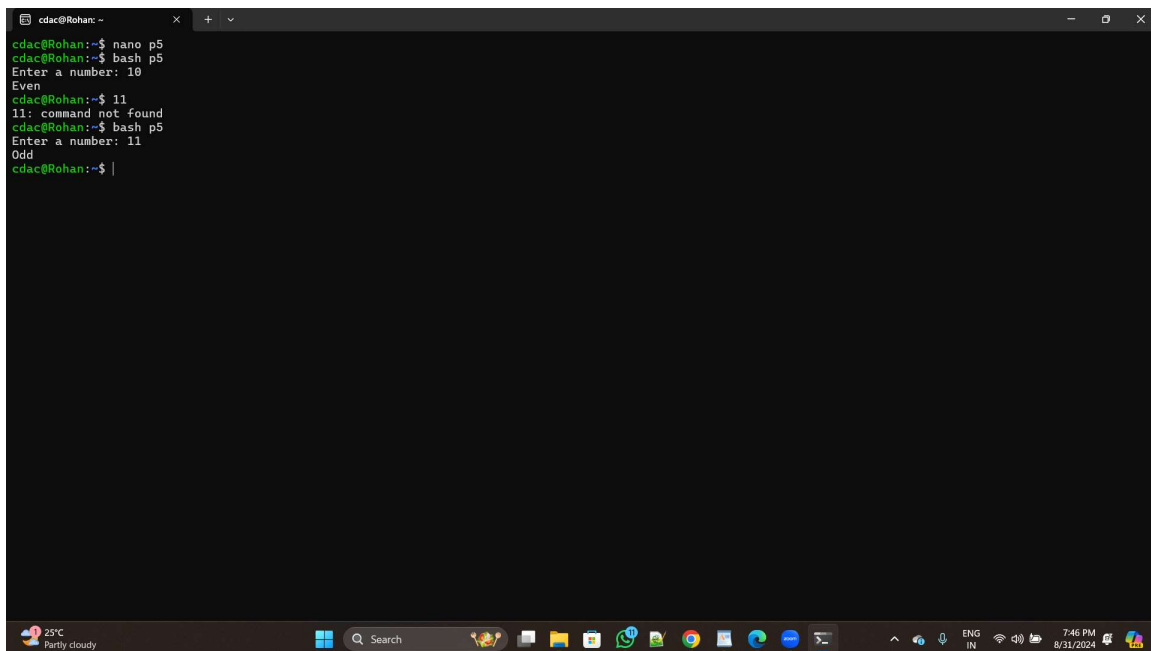
```
else
```

```
    echo "Odd"
```

```
fi
```

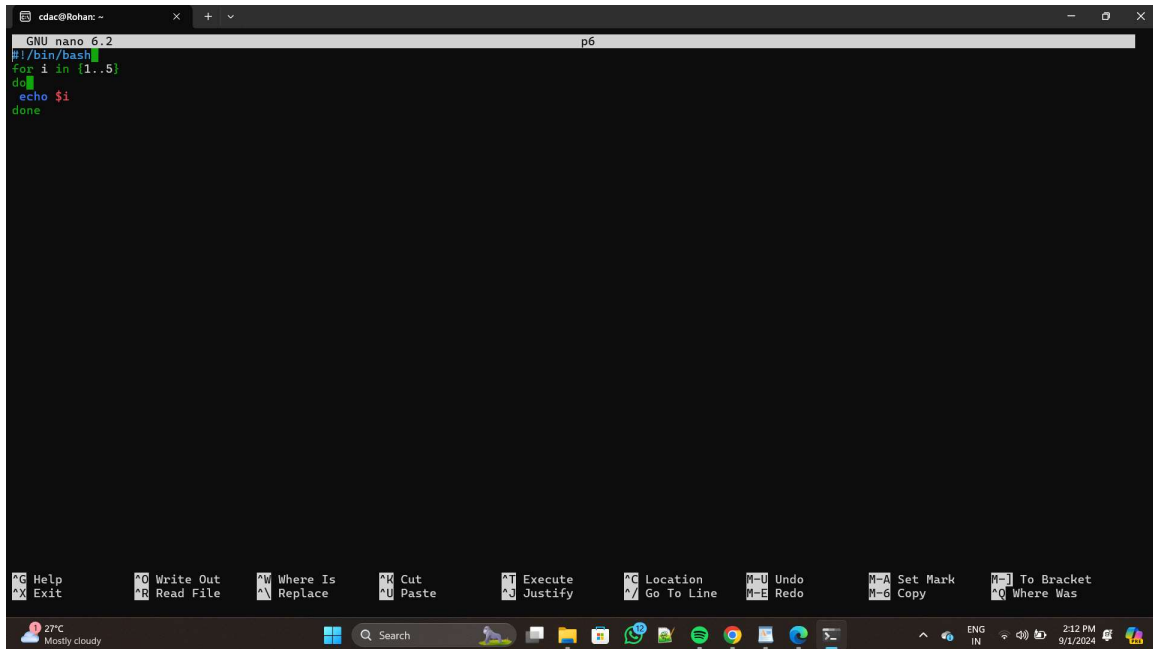
```
bash p5
```

then we get output as per our input.



```
cdac@Rohan: ~  
cdac@Rohan:~$ nano p5  
cdac@Rohan:~$ bash p5  
Enter a number: 10  
Even  
cdac@Rohan:~$ 11  
11: command not found  
cdac@Rohan:~$ bash p5  
Enter a number: 11  
Odd  
cdac@Rohan:~$ |
```

The screenshot shows a Windows terminal window titled 'cdac@Rohan: ~'. The user runs 'nano p5' to create a script, then 'bash p5' to execute it. The script prompts for a number. When '10' is entered, it outputs 'Even'. When '11' is entered, it outputs 'Odd'. The Windows taskbar at the bottom shows the date as 8/31/2024 and time as 7:46 PM.

A screenshot of a terminal window with a dark background. The window title is 'cdac@Rohanc: ~'. The terminal shows the GNU nano 6.2 editor with a file named 'p6'. The script content is:

```
#!/bin/bash
for i in {1..5}
do
  echo $i
done
```

 The bottom of the window features a menu bar with options like Help, Exit, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Location, Go To Line, Undo, Redo, Set Mark, Copy, To Bracket, and Where Was. Below the menu is a taskbar with various application icons and system status information including temperature (27°C), weather (Mostly cloudy), and time (2:12 PM 9/1/2024).

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

-->> nano p6

```
#!/bin/bash
```

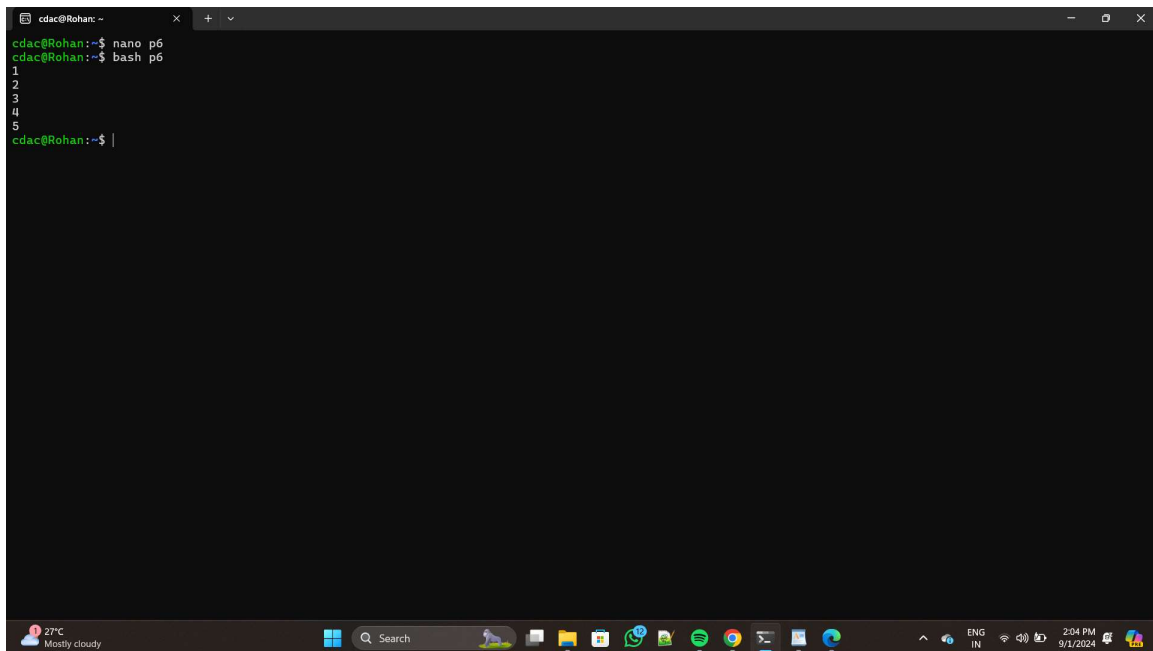
```
for i in {1..5}
```

do

echo \$i

done

bash p6

A screenshot of a Windows terminal window titled 'cdac@Rohan: ~'. The terminal shows a sequence of commands and their outputs. First, 'cdac@Rohan:~\$ nano p6' is entered. Then, 'cdac@Rohan:~\$ bash p6' is entered, which triggers the execution of a shell script. The script's output is displayed as a list of numbers: '1', '2', '3', '4', and '5', each on a new line. The prompt returns to 'cdac@Rohan:~\$' after the script finishes. The Windows taskbar is visible at the bottom, showing the date as 9/1/2024 and the time as 2:04 PM.

```
cdac@Rohan:~$ nano p6
cdac@Rohan:~$ bash p6
1
2
3
4
5
cdac@Rohan:~$
```

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

```
-->> nano p7
```

```
#!/bin/bash
```

```
i=1
```

```
while [ $i -le 5]
```

```
do
```

```
echo $i
```

```
i=$(( i + 1 ))
```

```
bash p7
```

```
cdac@Rohan: ~  
cdac@Rohan:~$ nano p7  
1  
2  
3  
4  
5  
cdac@Rohan:~$ nano p7
```

```
GNU nano 6.2 p7  
#!/bin/bash  
i=1  
while [ $i -le 5 ]  
do  
    echo $i  
    i=$((i + 1))  
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".

```
-->> nano p8
```

```
#!/bin/bash
```

```
if [ -f "file.txt" ]; then
```

```
echo "File exists"
```

```
else
```

```
echo "File does not exists"
```

```
fi
```

```
bash p8
```

```
GNU nano 6.2 p8
#!/bin/bash
if [ -f "File.txt" ];
then
echo "File Exists"
else
echo "File Does not exist"
fi
```

Help Exit Write Out Read File Where Is Replace Cut Paste Read 7 lines Execute Justify Location Go To Line Undo Redo Set Mark Copy To Bracket Where Was

```
cdac@Rohan:~$ nano p8
cdac@Rohan:~$ bash p8
File Does not exist
cdac@Rohan:~$ nano file.txt
cdac@Rohan:~$ bash p8
File Exists
cdac@Rohan:~$
```

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.

```
-->> nano p9
```

```
#!/bin/bash
```

```
read -p "Enter a number: " number
```

```
if [ $number -gt 10 ]; then
```

```
echo "The number entered is greater than 10"
```

```
else
```

```
echo "The number entered is less than 10"
```



```
GNU nano 6.2 p9
#!/bin/bash
read -p "Enter a number: " number
if [ $number -gt 10 ]; then
    echo "The number entered is greater than 10"
else
    echo "The number entered is less than 10"
fi
```

Help Exit Write Out Read File Where Is Replace Cut Paste Execute Justify Location Go To Line Undo Redo Set Mark Copy To Bracket Where Was

27°C Mostly cloudy 2:33 PM 9/1/2024

```
cdac@Rohan:~$ nano p9
cdac@Rohan:~$ bash p9
Enter a number: 11
The number entered is greater than 10
cdac@Rohan:~$ 9
9: command not found
cdac@Rohan:~$ bash p9
Enter a number: 9
The number entered is less than 10
cdac@Rohan:~$ |
```

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.

```
-->> nano p10
```

```
#!/bin/bash
```

```
for((i=1;i<=5;i++))
```

```
do
```

```
    for((j=1;j<=10;j++))
```

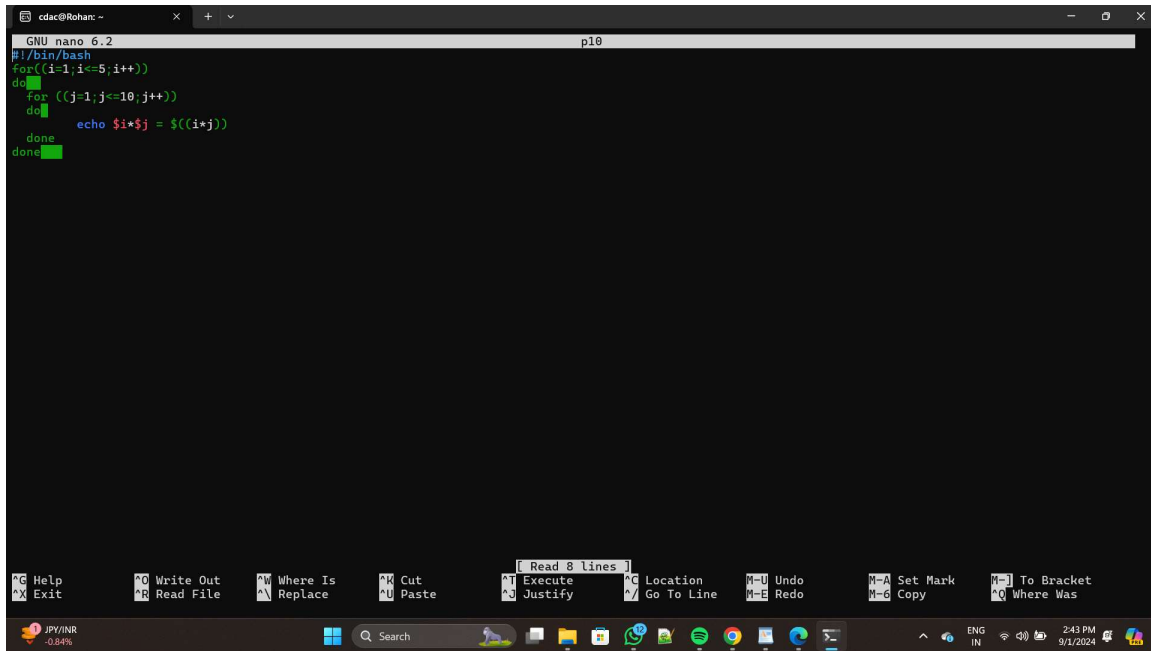
```
    do
```

```
        echo $i*$j = $((i*j))
```

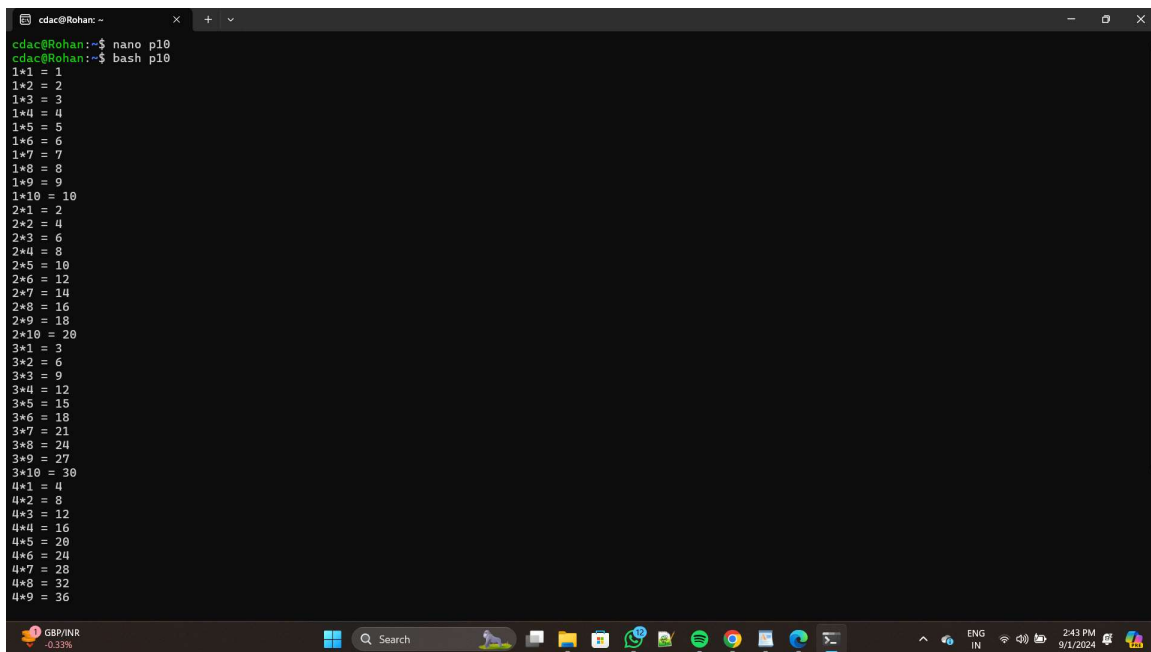
```
    done
```

```
done
```

bash p10



```
GNU nano 6.2 p10
#!/bin/bash
for((i=1;i<=5;i++))
do
  for((j=1;j<=10;j++))
  do
    echo $i*$j = ${i*j}
  done
done
```



```
cdac@Rohan:~$ nano p10
cdac@Rohan:~$ bash p10
1*1 = 1
1*2 = 2
1*3 = 3
1*4 = 4
1*5 = 5
1*6 = 6
1*7 = 7
1*8 = 8
1*9 = 9
1*10 = 10
2*1 = 2
2*2 = 4
2*3 = 6
2*4 = 8
2*5 = 10
2*6 = 12
2*7 = 14
2*8 = 16
2*9 = 18
2*10 = 20
3*1 = 3
3*2 = 6
3*3 = 9
3*4 = 12
3*5 = 15
3*6 = 18
3*7 = 21
3*8 = 24
3*9 = 27
3*10 = 30
4*1 = 4
4*2 = 8
4*3 = 12
4*4 = 16
4*5 = 20
4*6 = 24
4*7 = 28
4*8 = 32
4*9 = 36
```

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.

```
-->> nano p11
```

```
#!/bin/bash
```

```
while((1==1))
```

```
do
```

```
echo Enter a number
```

```
read n
```

```
if(($n>=0))
```

```
then
```

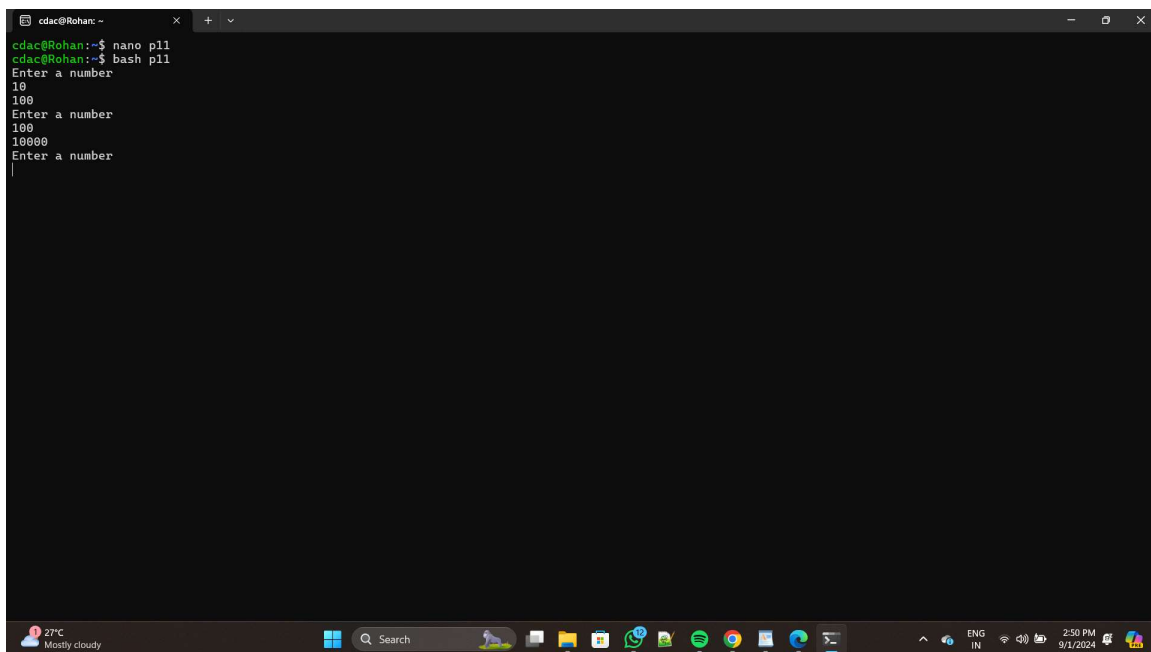
`echo $((n*n))`

`else`

`break`

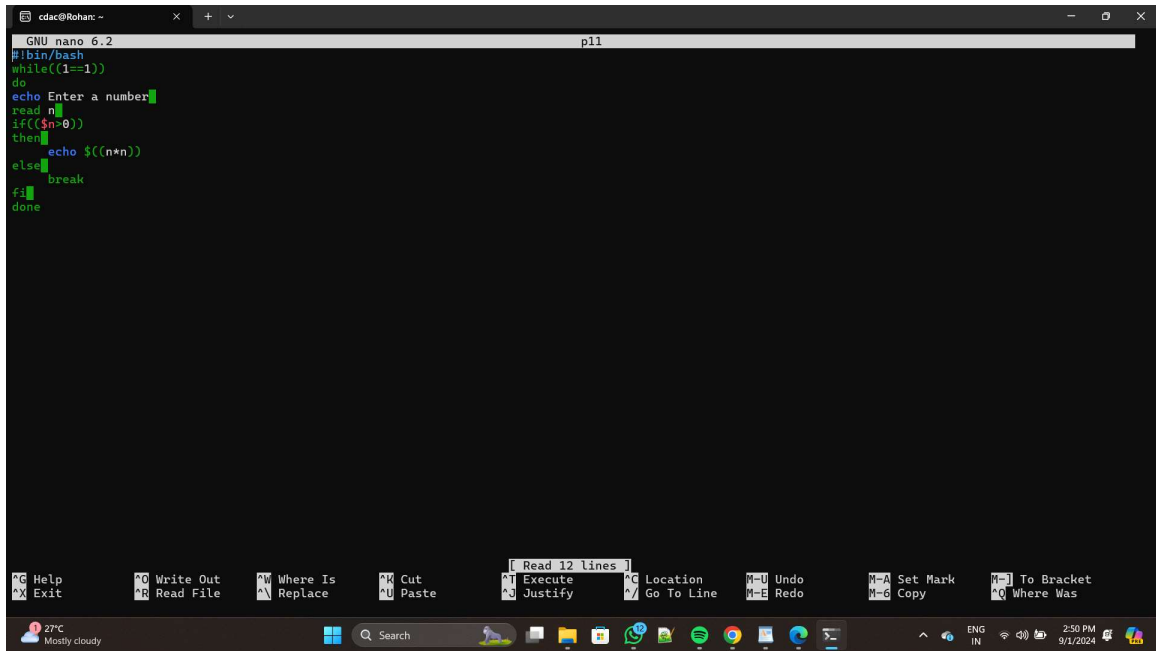
`fi`

`done`



```
cdac@Rohan: ~  
cdac@Rohan:~$ nano p11  
cdac@Rohan:~$ bash p11  
Enter a number  
10  
100  
Enter a number  
100  
10000  
Enter a number  
|
```

The screenshot shows a Windows terminal window titled 'cdac@Rohan: ~'. The user has created a file named 'p11' using 'nano' and then executed it with 'bash p11'. The script prompts the user to 'Enter a number' three times. The first input is '10', the second is '100', and the third is '10000'. The cursor is currently on the line following the third prompt. The Windows taskbar is visible at the bottom, showing the time as 2:58 PM on 9/1/2024.



The screenshot shows a Windows 11 desktop environment. A terminal window titled "cdac@Rohanc ~" is open, displaying the GNU nano 6.2 editor. The editor is editing a file named "p11". The script content is as follows:

```
#!/bin/bash
while((1==1))
do
echo Enter a number
read n
if((n>0))
then
echo $((n*n))
else
break
fi
done
```

The nano editor's status bar at the bottom shows "Read 12 lines". The Windows taskbar at the bottom displays the system tray with the date and time as "2:58 PM 9/1/2024".