Assignment -2 COS

Part A

What will the following commands do?

echo "Hello, World!"

It will print Hello World! as it is.

name="Productive”
It assigns the string value
`"Productive"` to the variable named `name`.

touch file.txt

It will create `file.txt` file.

ls -a

Lists all files and directories in the current working directory.

rm file.txt

This will remove file `file.txt`

cp file1.txt file2.txt
The command
`cp file1.txt file2.txt` copies the contents of `file1.txt` into a new file named
`file2.txt`.

mv file.txt /path/to/directory/
The command
`mv file.txt /path/to/directory/` moves the file `file.txt` from its current location to the directory specified by `/path/to/directory/`.

chmod 755
script.sh

Changes the permissions of `script.sh` so that: The owner can read, write, and execute. The group and others can read and execute, but not write.

grep "pattern" file.txt
The command will search through
`file.txt` and print out all lines that contain the specified `"pattern"`.

kill PID

This command will terminate/stop the current process

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

This will create a directory `mydir` and the inside that it will create a file `file1.txt` then inside file1.txt it will write Hello World! then also it will display all the file content on the terminal screen.

ls -l | grep ".txt"
The command
`ls -l | grep ".txt"` lists detailed information about files in the current directory and filters the output to show only files with a `.txt` extension.

cat file1.txt file2.txt | sort | uniq
The command
`cat file1.txt file2.txt | sort | uniq` combines the contents of `file1.txt` and `file2.txt`, sorts the lines, and removes any duplicates.

ls -l | grep "^d"
The command
`ls -l | grep "^d"` lists only directories in the current directory.

grep -r "pattern" /path/to/directory/

The command `grep -r "pattern" /path/to/directory/` searches for the specified pattern in all files within the given directory and its subdirectories.

cat file1.txt file2.txt | sort | uniq –d

It is used to find and display duplicate lines that appear in both `file1.txt` and `file2.txt`.

chmod 644 file.txt

`chmod 644 file.txt` Sets file permissions to allow the owner to read and write, and others (group and everyone else) to read only.

cp -r source_directory destination_directory

The command `cp -r source_directory destination_directory` copies the entire contents of `source_directory` into `destination_directory`.

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

The command `find /path/to/search -name "*.txt"` searches for files with a `.txt` extension within the specified directory and its subdirectories.

chmod u+x file.txt

The command `chmod u+x file.txt` adds execute permission for the owner of `file.txt`

echo $PATH

The command `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.

   True

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.

   chmod

2. cpy is used to copy files and directories.

   cp

3. mkfile is used to create a new file.

   touch

4. catx is used to concatenate files.

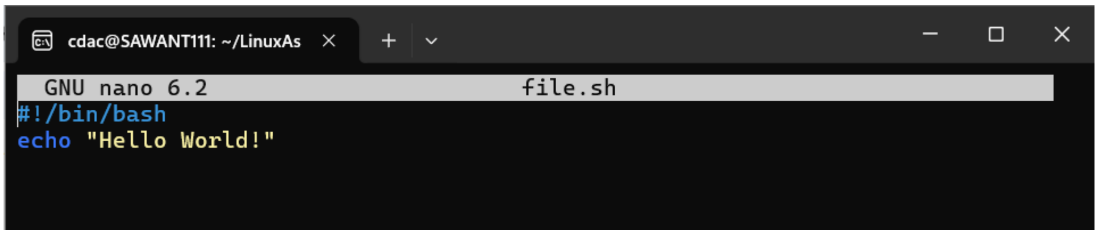
   cat

5. rn is used to rename files.

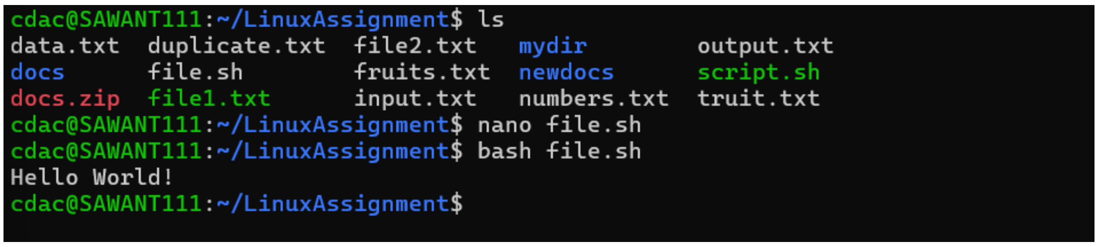
mv

Part C

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

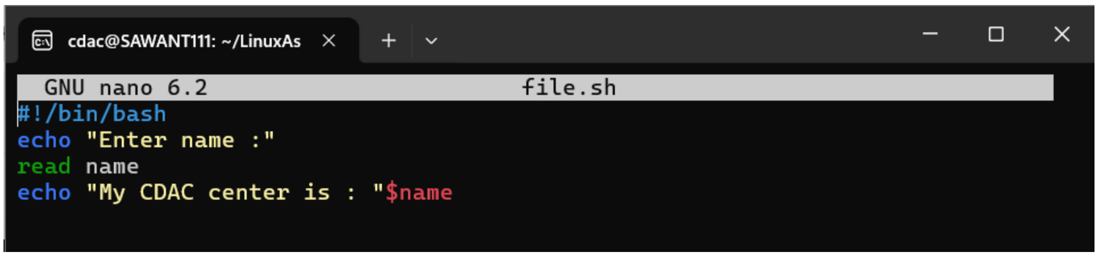


```
  GNU nano 6.2                          file.sh
#!/bin/bash
echo "Hello World!"
```

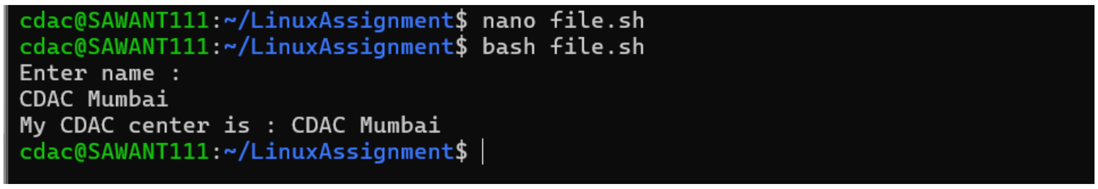


```
cdac@SAWANT111:~/LinuxAssignment$ ls
data.txt  duplicate.txt  file2.txt   mydir        output.txt
docs      file.sh        fruits.txt  newdocs      script.sh
docs.zip  file1.txt      input.txt   numbers.txt  truit.txt
cdac@SAWANT111:~/LinuxAssignment$ nano file.sh
cdac@SAWANT111:~/LinuxAssignment$ bash file.sh
Hello World!
cdac@SAWANT111:~/LinuxAssignment$
```

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

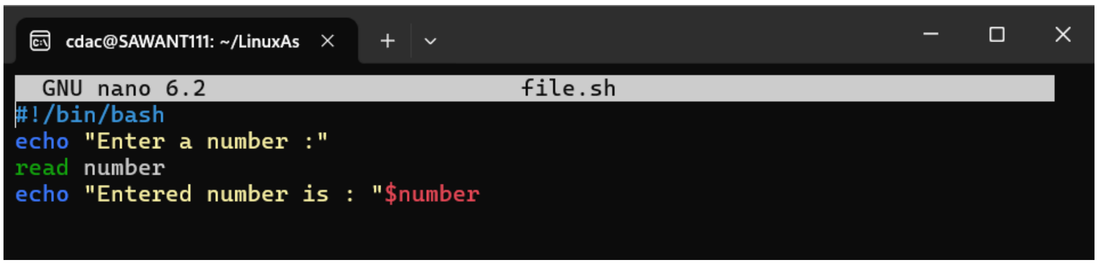


```
  GNU nano 6.2                          file.sh
#!/bin/bash
echo "Enter name :"
read name
echo "My CDAC center is : "$name
```

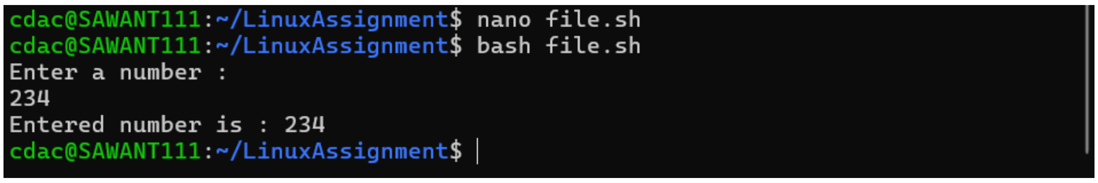


```
cdac@SAWANT111:~/LinuxAssignment$ nano file.sh
cdac@SAWANT111:~/LinuxAssignment$ bash file.sh
Enter name :
CDAC Mumbai
My CDAC center is : CDAC Mumbai
cdac@SAWANT111:~/LinuxAssignment$ 
```

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

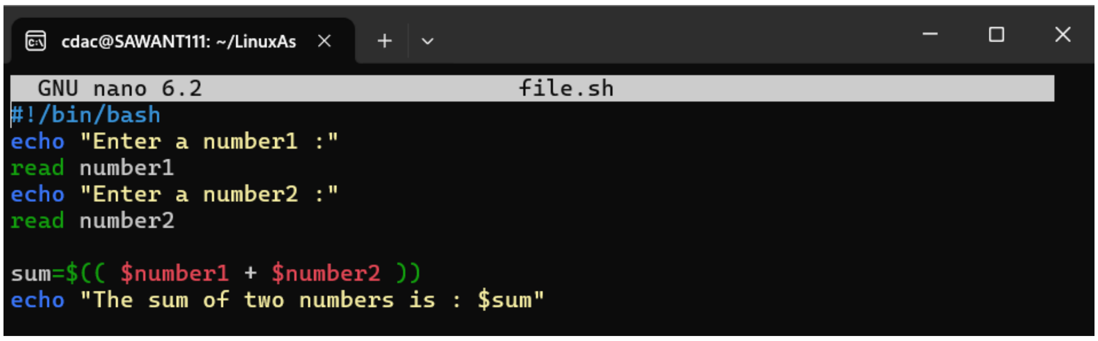


```
  GNU nano 6.2                              file.sh
#!/bin/bash
echo "Enter a number :"
read number
echo "Entered number is : "$number
```

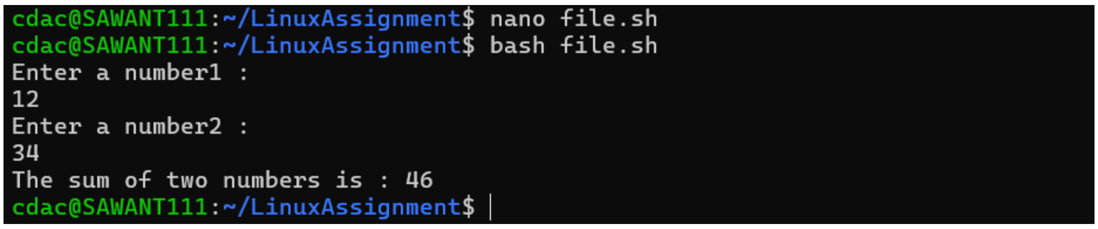


```
cdac@SAWANT111:~/LinuxAssignment$ nano file.sh
cdac@SAWANT111:~/LinuxAssignment$ bash file.sh
Enter a number :
234
Entered number is : 234
cdac@SAWANT111:~/LinuxAssignment$
```

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

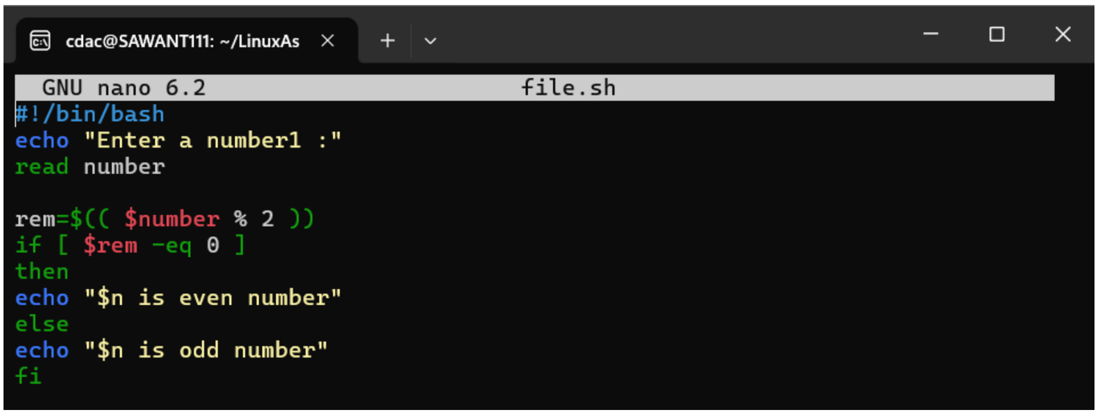


```
  GNU nano 6.2                              file.sh
#!/bin/bash
echo "Enter a number1 :"
read number1
echo "Enter a number2 :"
read number2

sum=$(( $number1 + $number2 ))
echo "The sum of two numbers is : $sum"
```

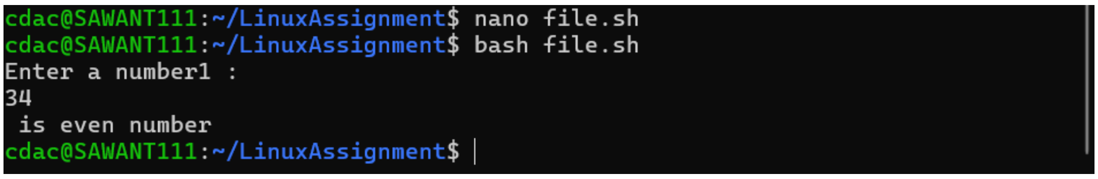


```
cdac@SAWANT111:~/LinuxAssignment$ nano file.sh
cdac@SAWANT111:~/LinuxAssignment$ bash file.sh
Enter a number1 :
12
Enter a number2 :
34
The sum of two numbers is : 46
cdac@SAWANT111:~/LinuxAssignment$
```

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

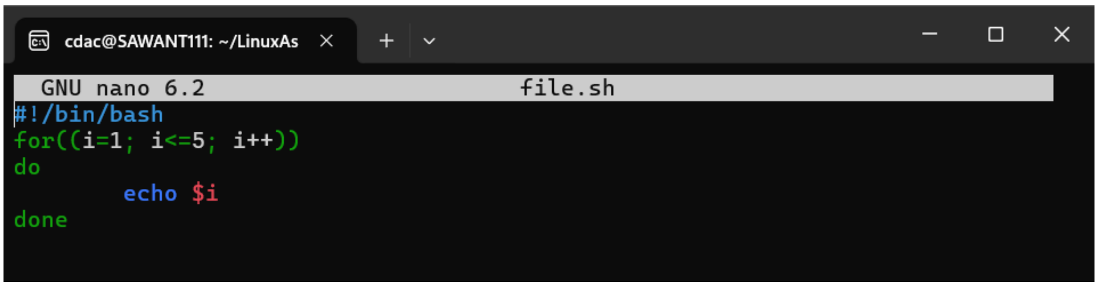


```
  GNU nano 6.2                                file.sh
#!/bin/bash
echo "Enter a number1 :"
read number

rem=$(( $number % 2 ))
if [ $rem -eq 0 ]
then
echo "$n is even number"
else
echo "$n is odd number"
fi
```

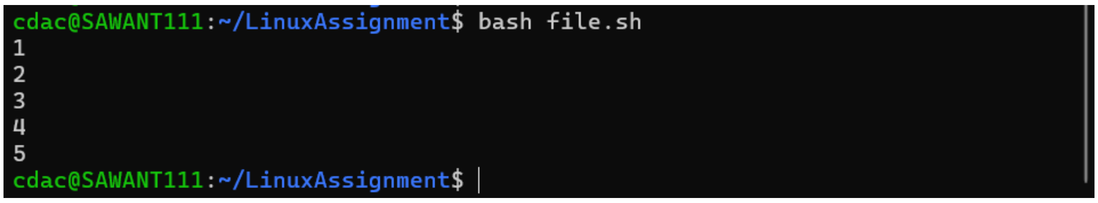


```
cdac@SAWANT111:~/LinuxAssignment$ nano file.sh
cdac@SAWANT111:~/LinuxAssignment$ bash file.sh
Enter a number1 :
34
 is even number
cdac@SAWANT111:~/LinuxAssignment$
```

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

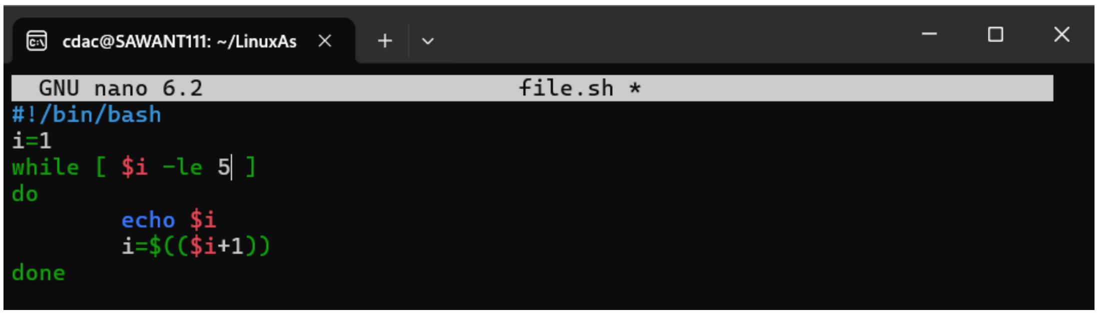


```
  GNU nano 6.2                                file.sh
#!/bin/bash
for((i=1; i<=5; i++))
do
        echo $i
done
```

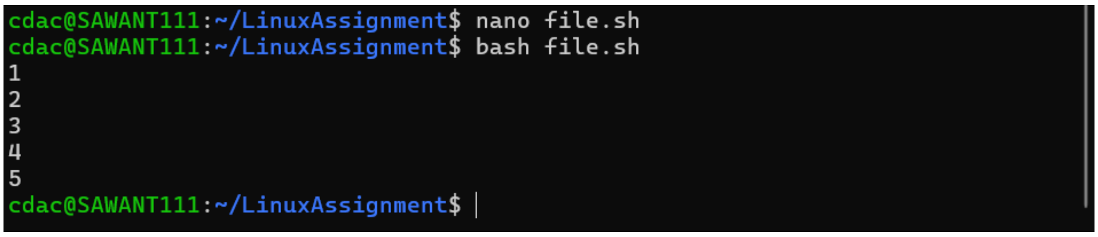


```
cdac@SAWANT111:~/LinuxAssignment$ bash file.sh
1
2
3
4
5
cdac@SAWANT111:~/LinuxAssignment$
```

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

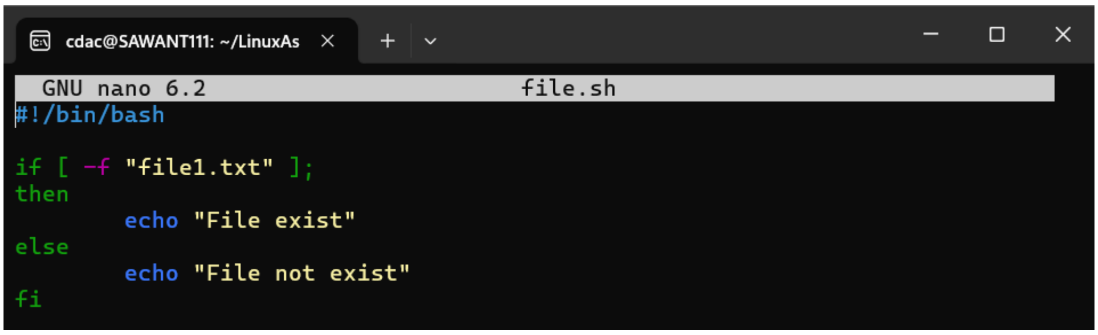


```
cdac@SAWANT111: ~/LinuxAs
 GNU nano 6.2                        file.sh *
#!/bin/bash
i=1
while [ $i -le 5 ]
do
        echo $i
        i=$(($i+1))
done
```

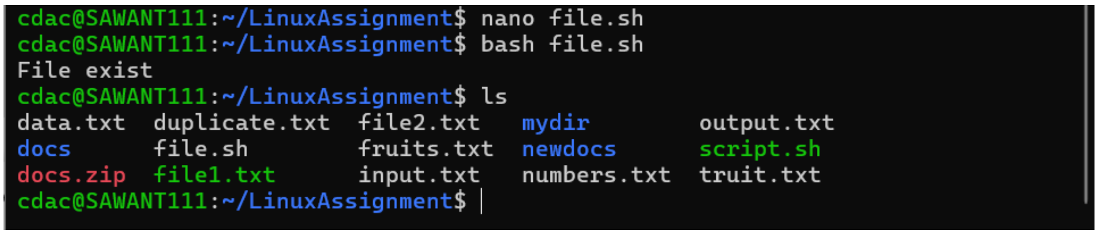


```
cdac@SAWANT111:~/LinuxAssignment$ nano file.sh
cdac@SAWANT111:~/LinuxAssignment$ bash file.sh
1
2
3
4
5
cdac@SAWANT111:~/LinuxAssignment$
```

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".

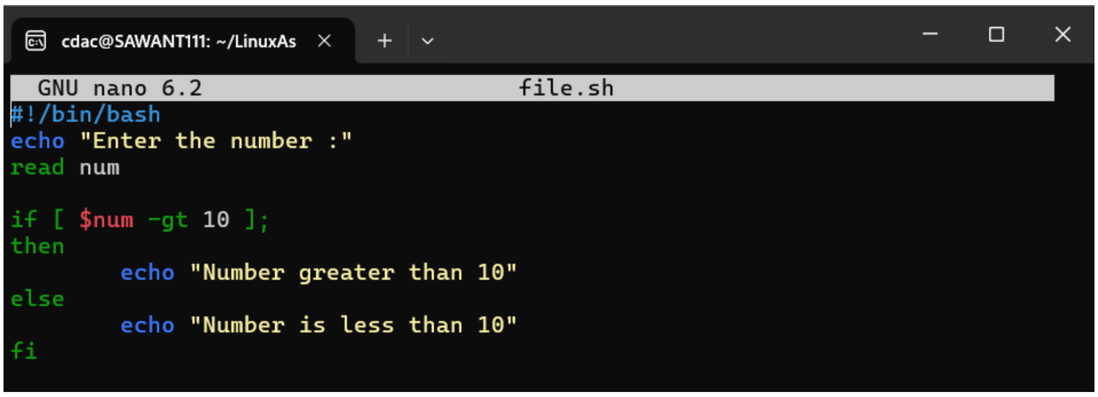


```
cdac@SAWANT111: ~/LinuxAs
 GNU nano 6.2                        file.sh
#!/bin/bash

if [ -f "file1.txt" ];
then
        echo "File exist"
else
        echo "File not exist"
fi
```

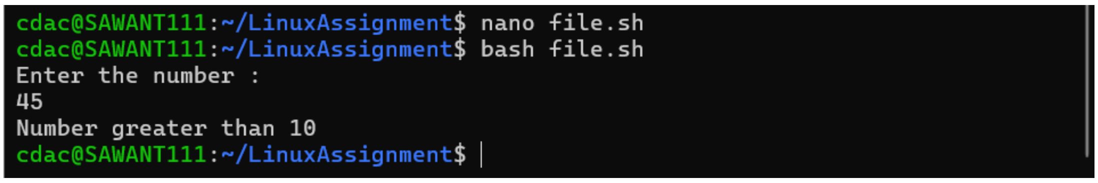


```
cdac@SAWANT111:~/LinuxAssignment$ nano file.sh
cdac@SAWANT111:~/LinuxAssignment$ bash file.sh
File exist
cdac@SAWANT111:~/LinuxAssignment$ ls
data.txt  duplicate.txt  file2.txt   mydir        output.txt
docs      file.sh        fruits.txt  newdocs      script.sh
docs.zip  file1.txt      input.txt   numbers.txt  truit.txt
cdac@SAWANT111:~/LinuxAssignment$
```

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.

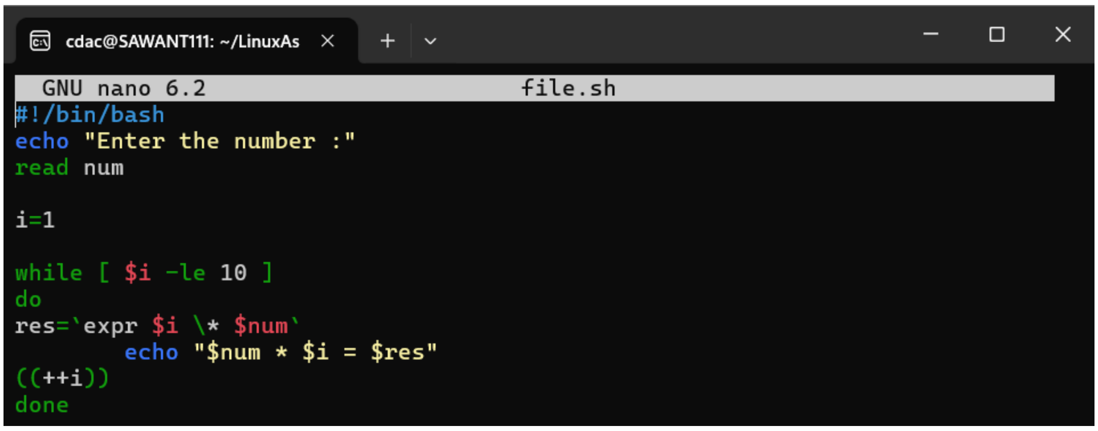


```
  GNU nano 6.2                            file.sh
#!/bin/bash
echo "Enter the number :"
read num

if [ $num -gt 10 ];
then
        echo "Number greater than 10"
else
        echo "Number is less than 10"
fi
```

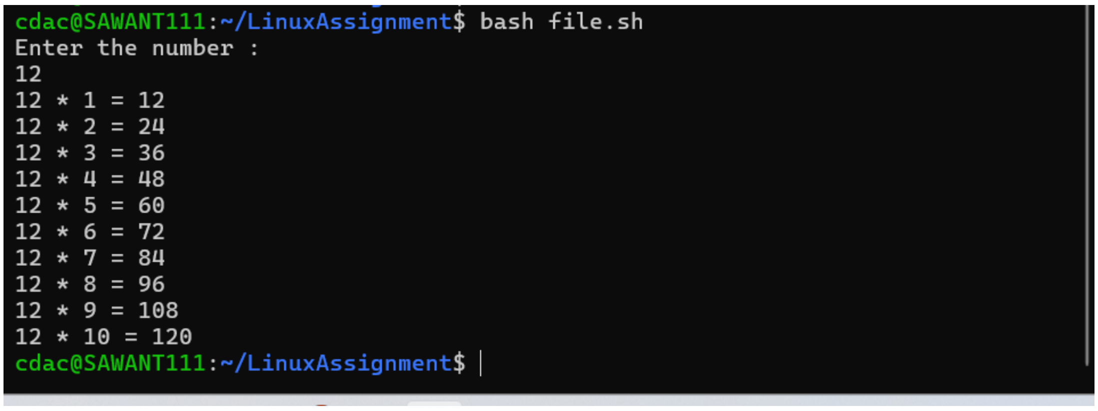


```
cdac@SAWANT111:~/LinuxAssignment$ nano file.sh
cdac@SAWANT111:~/LinuxAssignment$ bash file.sh
Enter the number :
45
Number greater than 10
cdac@SAWANT111:~/LinuxAssignment$ 
```

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.

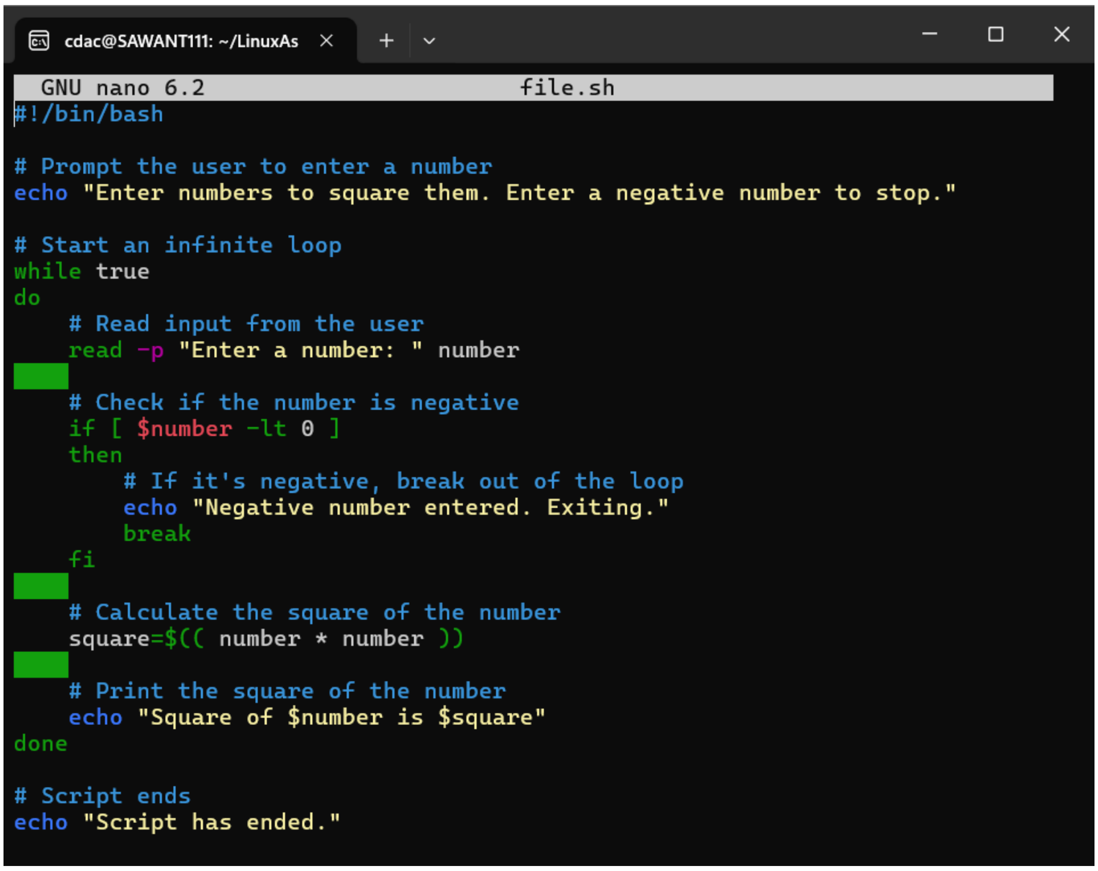


```
  GNU nano 6.2                            file.sh
#!/bin/bash
echo "Enter the number :"
read num

i=1

while [ $i -le 10 ]
do
res=`expr $i \* $num`
        echo "$num * $i = $res"
((++i))
done
```

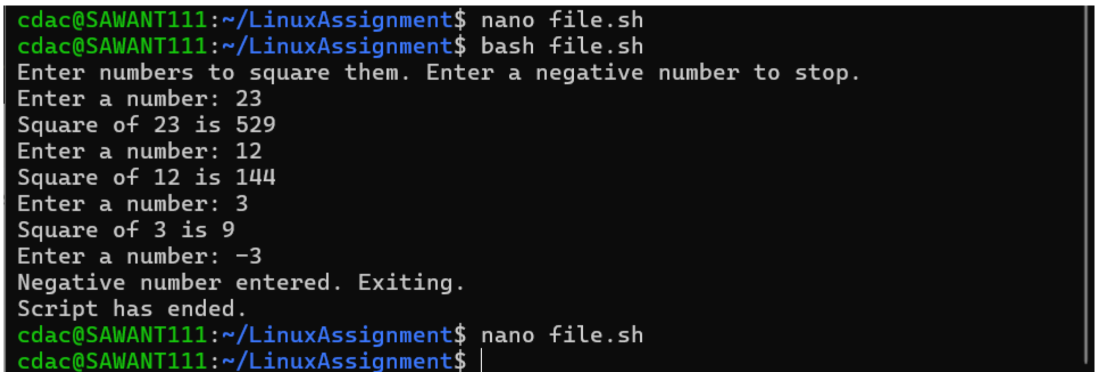


```
cdac@SAWANT111:~/LinuxAssignment$ bash file.sh
Enter the number :
12
12 * 1 = 12
12 * 2 = 24
12 * 3 = 36
12 * 4 = 48
12 * 5 = 60
12 * 6 = 72
12 * 7 = 84
12 * 8 = 96
12 * 9 = 108
12 * 10 = 120
cdac@SAWANT111:~/LinuxAssignment$ 
```

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.



```
  GNU nano 6.2                               file.sh
#!/bin/bash

# Prompt the user to enter a number
echo "Enter numbers to square them. Enter a negative number to stop."

# Start an infinite loop
while true
do
    # Read input from the user
    read -p "Enter a number: " number

    # Check if the number is negative
    if [ $number -lt 0 ]
    then
        # If it's negative, break out of the loop
        echo "Negative number entered. Exiting."
        break
    fi

    # Calculate the square of the number
    square=$(( number * number ))

    # Print the square of the number
    echo "Square of $number is $square"
done

# Script ends
echo "Script has ended."
```



```
cdac@SAWANT111:~/LinuxAssignment$ nano file.sh
cdac@SAWANT111:~/LinuxAssignment$ bash file.sh
Enter numbers to square them. Enter a negative number to stop.
Enter a number: 23
Square of 23 is 529
Enter a number: 12
Square of 12 is 144
Enter a number: 3
Square of 3 is 9
Enter a number: -3
Negative number entered. Exiting.
Script has ended.
cdac@SAWANT111:~/LinuxAssignment$ nano file.sh
cdac@SAWANT111:~/LinuxAssignment$ 
```

COS Assignment - 2

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.

→

① Completion Time →

P1
P2
P3

Process	AT	BT	Completion Time	
P1	0	5	5	(0+5)
P2	1	3	8	(3+5)
P3	2	6	14	(8+6)

② TAT = CT − AT (Turn around Time).

Process	AT	BT	CT	TAT	
P1	0	5	5	5	(5−0)
P2	1	3	8	7	(8−1).
P3	2	6	14	12	(14−2)

③ Waiting time WT = TAT − BT

P1
P2
P3

Process	AT	BT	CT	TAT	WT	
P1	0	5	5	5	0	(5−5)
P2	1	3	8	7	4	(7−3)
P3	2	6	14	12	6	(12−6).

Page No :
Date:

$$\text{Avg waiting time} = \frac{0+4+6}{3} = \frac{10}{3}$$

$= 3.33$ 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.

→ Order the processes based on their burst time :
Execution order →

$P1 \to P3 \to P4 \to P2.$ TAT = CT − AT

(0→3) (3→4) (4→8) (8→13)

Process	AT	BT	CT	T.AT
P1	0	3	3	3
P3	2	1	4	2
P4	3	4	8	5
P2	1	5	13	12

$$\text{Avg. Turn Around Time} = \frac{3+2+5+12}{4}$$

$$= \frac{22}{4}$$

$= 5.5$ units

3. Consider the following processes with arrival times, burst times, and priorities (lower number indicates higher priority):
Calculate the average waiting time using Priority Scheduling.

Process	Arrival Time	Burst Time	Priority
P1	0	6	3
P2	1	4	1
P3	2	7	4
P4	3	2	2

→

$$P1 \longrightarrow P2 \longrightarrow P4 \longrightarrow P3.$$
(0-6) (6-10) (10-12) (12-19).

Process	AT	BT	CT	TAT	WT
P1	0	6	6	(6-0) 6	(6-6) 0
P2	1	4	10	(10-1) 9	(9-4) 5
P4	3	2	12	(12-3) 9	(9-2) 7
P3	2	7	19	(19-2) 17	(17-7) 10

$$\text{Avg WT} = \frac{0+5+7+10}{4}$$

$$= \frac{22}{4} = 5.5 \text{ units.}$$

4. Consider the following Process 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 avg turnaround time using Round Robin Scheduling.