CDAC MUMBAI

Concepts of Operating System

Assignment 2

NAME - Sudnyan Pal_JH

Part A

What will the following commands do?

- echo "Hello, World!"
  - This will print the given text to the terminal.
- name="Productive"
  - This will set variable 'name' to the value 'Productive'.
- touch file.txt
  - This creates an empty file named file.txt
- ls -a
  - It lists all files and directories in the current directory, including hidden files
- rm file.txt
  - It removes the file.txt fron current directory.
- cp file1.txt file2.txt
  - It copies the content of file1.txt to file2.txt
- mv file.txt /path/to/directory/
  - It moves file.txt to given directory path
- chmod 755 script.sh
  - It changes the permission of script.sh so that owner has read,write and execute permision and everyone had read and execute permission.
- grep "pattern" file.txt
  - It finds the string pattern in file.txt and prints lines that contain word pattern.
- kill PID
  - This terminate the process with process id PID
- mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt
  - This creates directory mydir then change directory to mydir and create empty file file.txt and write "Hello, World!" into file.txt .
- ls -l | grep ".txt"
  - This command lists files in long format and then filters the output, showing only the files that have .txt in their names.
- cat file1.txt file2.txt | sort | uniq
  - Concatenates content of file1.txt and file2.txt and sort combined content and show unique lines.
- ls -l | grep "^d"
- - lists files in long format and then filters the output to show only directories
- grep -r "pattern" /path/to/directory/
  - It recursively searches for "pattern" in all files under given directory and prints the matching lines.
- cat file1.txt file2.txt | sort | uniq –d
  - It concatenates the content of file1.txt and file2.txt and sort content and display only duplicated lines.
- chmod 644 file.txt
  - It changes permission of file.txt to read and write for owner and read only for every one else.
- cp -r source_directory destination_directory
  - It copies entire source_directory and its subdirectories to destination_directory

CDAC Mumbai

- find /path/to/search -name "*.txt"
  - It searches for all files with the .txt extension under the specified path
- chmod u+x file.txt
  - It adds execute permission to user for file file.txt.
- echo $PATH
  - It will print current value of path environment variable.

Part B

Identify True or False:

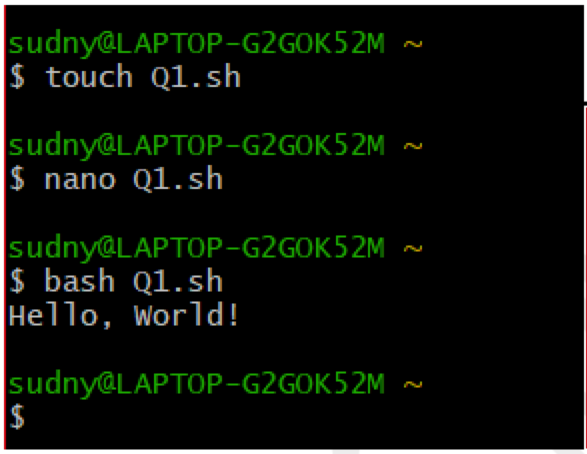
1. **ls** is used to list files and directories in a directory. **TRUE**
2. **mv** is used to move files and directories. **TRUE**
3. **cd** is used to copy files and directories. **FALSE, cp is used to copy files and directories**.
4. **pwd** stands for "print working directory" and displays the current 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**

Identify the Incorrect Commands:

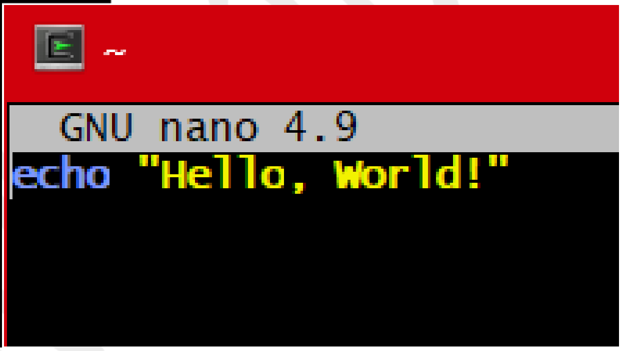
1. **chmodx** is used to change file permissions. **chmod**
2. **cpy** is used to copy files and directories. **cp**
3. **mkfile** is used to create a new file. **touch**
4. **catx** is used to concatenate files. **cat**
5. **rn** is used to rename files. **mv**

Part C

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

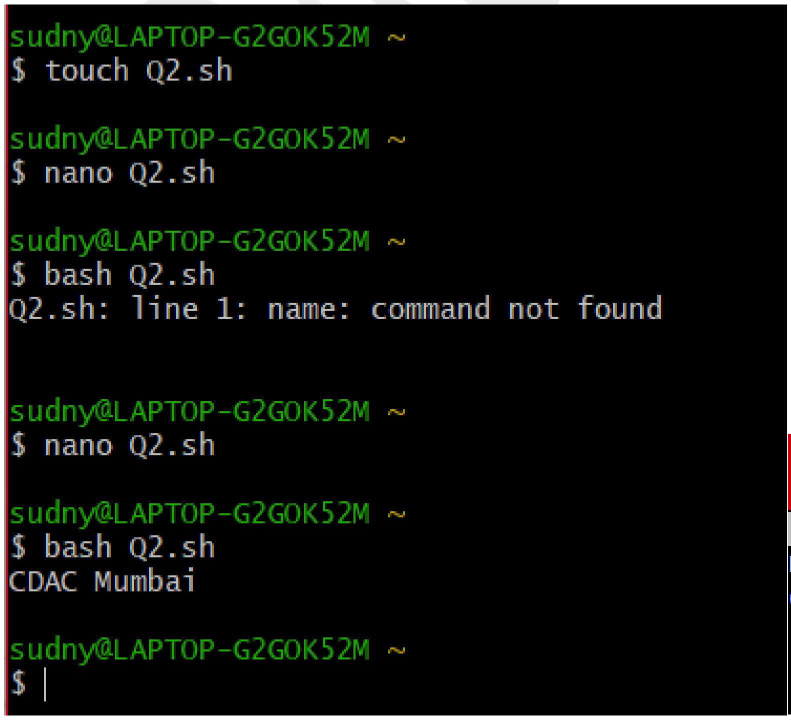


```
sudny@LAPTOP-G2GOK52M ~
$ touch Q1.sh

sudny@LAPTOP-G2GOK52M ~
$ nano Q1.sh

sudny@LAPTOP-G2GOK52M ~
$ bash Q1.sh
Hello, World!

sudny@LAPTOP-G2GOK52M ~
$
```

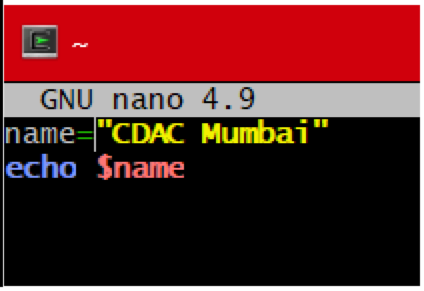


```
~
GNU nano 4.9
echo "Hello, World!"
```

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



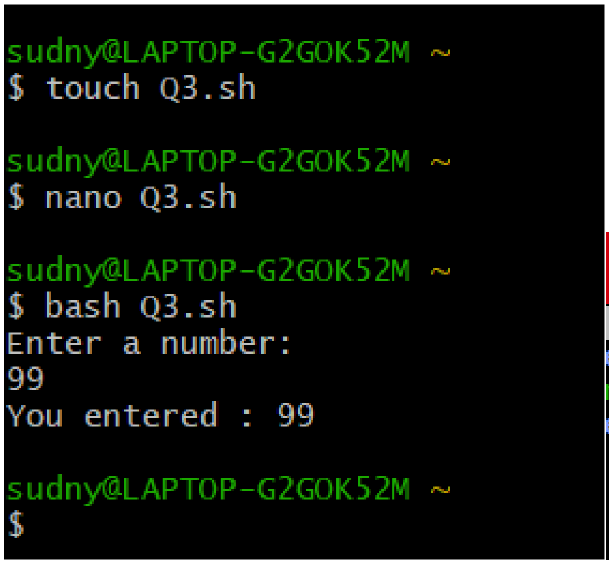
```
sudny@LAPTOP-G2GOK52M ~
$ touch Q2.sh

sudny@LAPTOP-G2GOK52M ~
$ nano Q2.sh

sudny@LAPTOP-G2GOK52M ~
$ bash Q2.sh
Q2.sh: line 1: name: command not found


sudny@LAPTOP-G2GOK52M ~
$ nano Q2.sh

sudny@LAPTOP-G2GOK52M ~
$ bash Q2.sh
CDAC Mumbai

sudny@LAPTOP-G2GOK52M ~
$ |
```



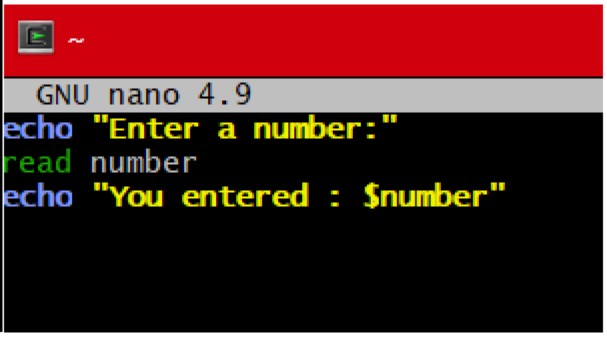
```
~
GNU nano 4.9
name="CDAC Mumbai"
echo $name
```

CDAC Mumbai

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

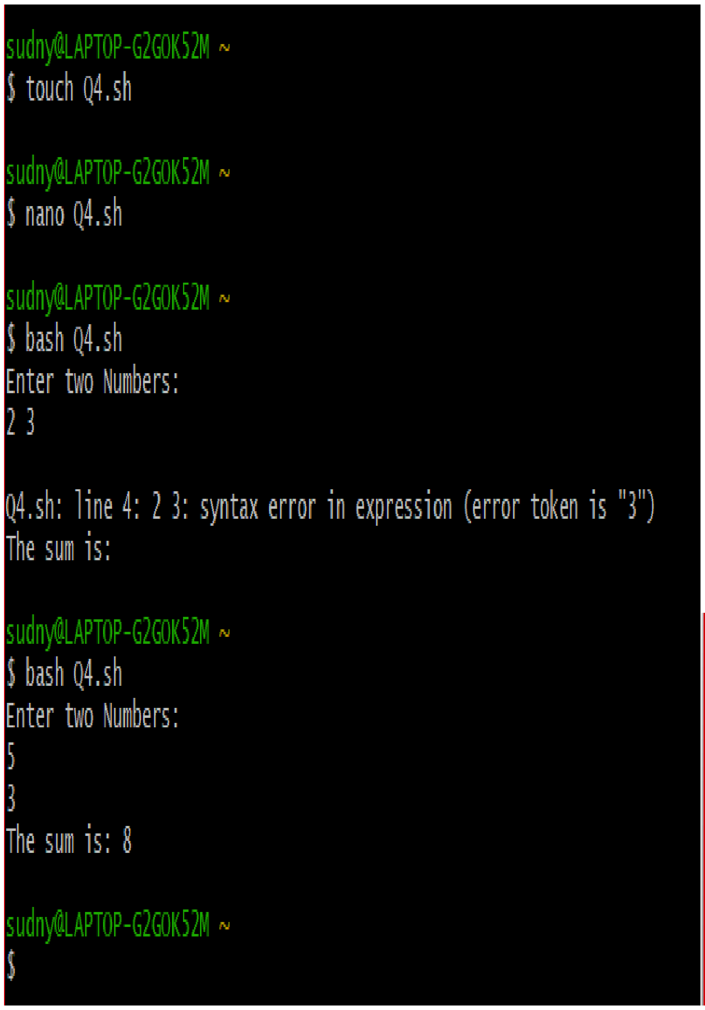


```
sudny@LAPTOP-G2GOK52M ~
$ touch Q3.sh

sudny@LAPTOP-G2GOK52M ~
$ nano Q3.sh

sudny@LAPTOP-G2GOK52M ~
$ bash Q3.sh
Enter a number:
99
You entered : 99

sudny@LAPTOP-G2GOK52M ~
$
```

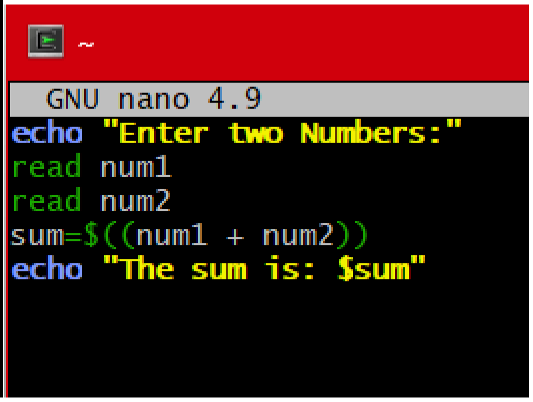


```
~
 GNU nano 4.9
echo "Enter a number:"
read number
echo "You entered : $number"
```

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

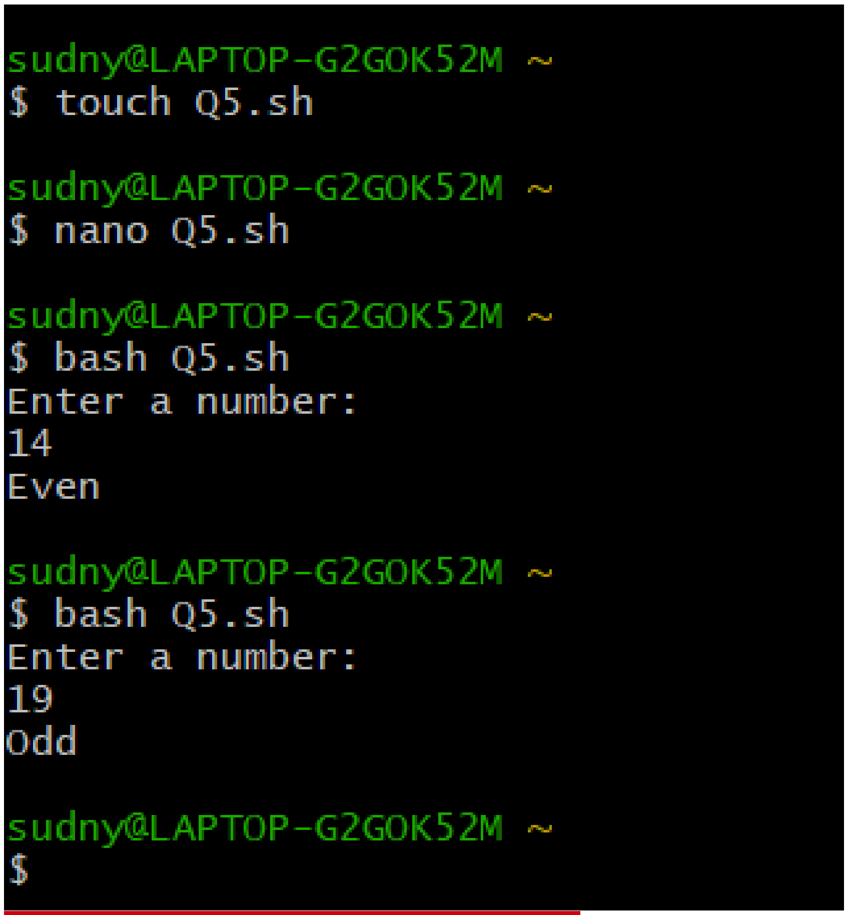


```
sudny@LAPTOP-G2GOK52M ~
$ touch Q4.sh

sudny@LAPTOP-G2GOK52M ~
$ nano Q4.sh

sudny@LAPTOP-G2GOK52M ~
$ bash Q4.sh
Enter two Numbers:
2 3

Q4.sh: line 4: 2 3: syntax error in expression (error token is "3")
The sum is:

sudny@LAPTOP-G2GOK52M ~
$ bash Q4.sh
Enter two Numbers:
5
3
The sum is: 8

sudny@LAPTOP-G2GOK52M ~
$
```

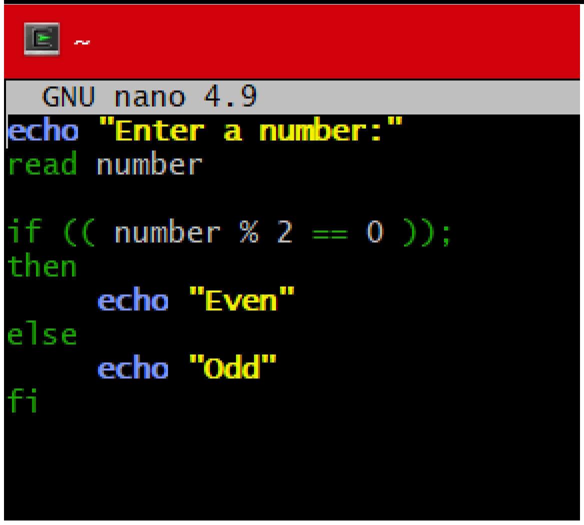


```
~
 GNU nano 4.9
echo "Enter two Numbers:"
read num1
read num2
sum=$((num1 + num2))
echo "The sum is: $sum"
```

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

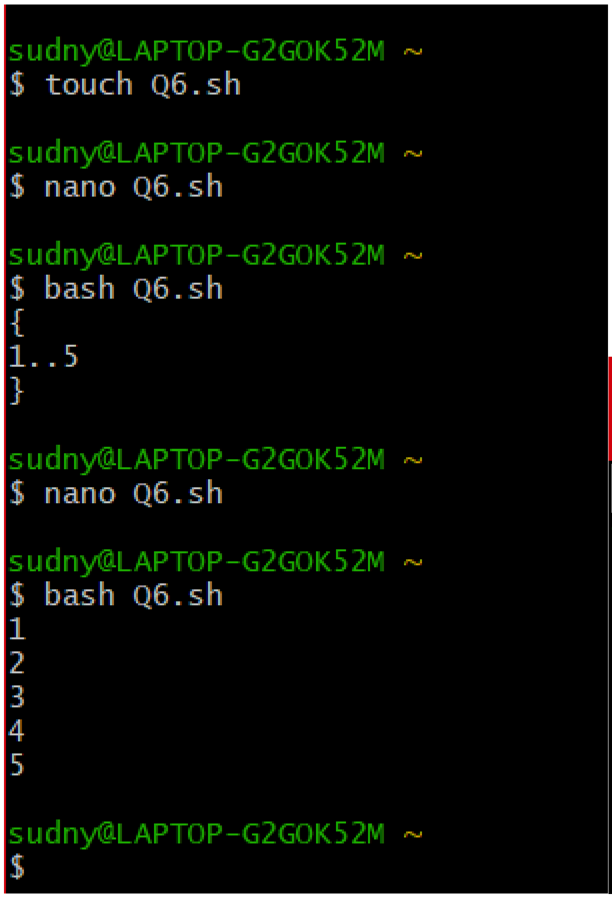


```
sudny@LAPTOP-G2GOK52M ~
$ touch Q5.sh

sudny@LAPTOP-G2GOK52M ~
$ nano Q5.sh

sudny@LAPTOP-G2GOK52M ~
$ bash Q5.sh
Enter a number:
14
Even

sudny@LAPTOP-G2GOK52M ~
$ bash Q5.sh
Enter a number:
19
Odd

sudny@LAPTOP-G2GOK52M ~
$
```

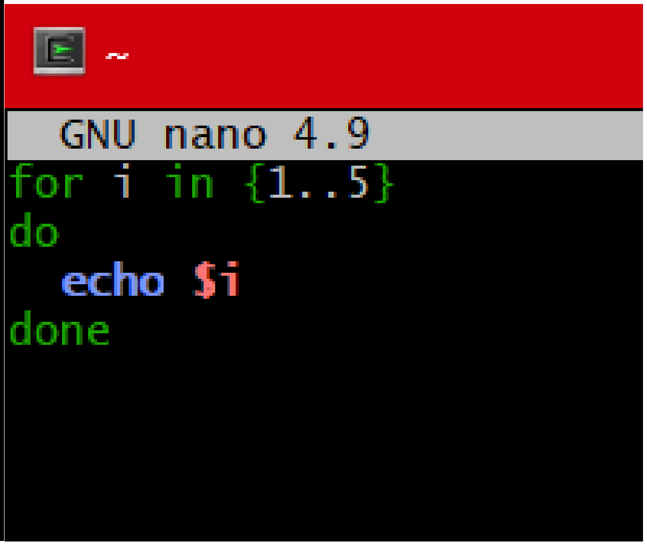


```
~
  GNU nano 4.9
echo "Enter a number:"
read number

if (( number % 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.

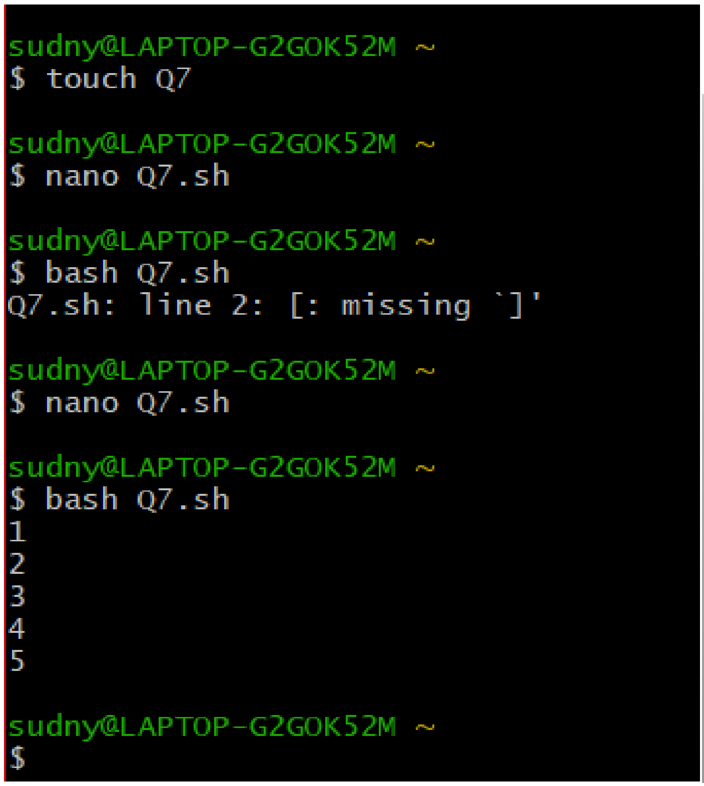


```
sudny@LAPTOP-G2GOK52M ~
$ touch Q6.sh

sudny@LAPTOP-G2GOK52M ~
$ nano Q6.sh

sudny@LAPTOP-G2GOK52M ~
$ bash Q6.sh
{
1..5
}

sudny@LAPTOP-G2GOK52M ~
$ nano Q6.sh

sudny@LAPTOP-G2GOK52M ~
$ bash Q6.sh
1
2
3
4
5

sudny@LAPTOP-G2GOK52M ~
$
```

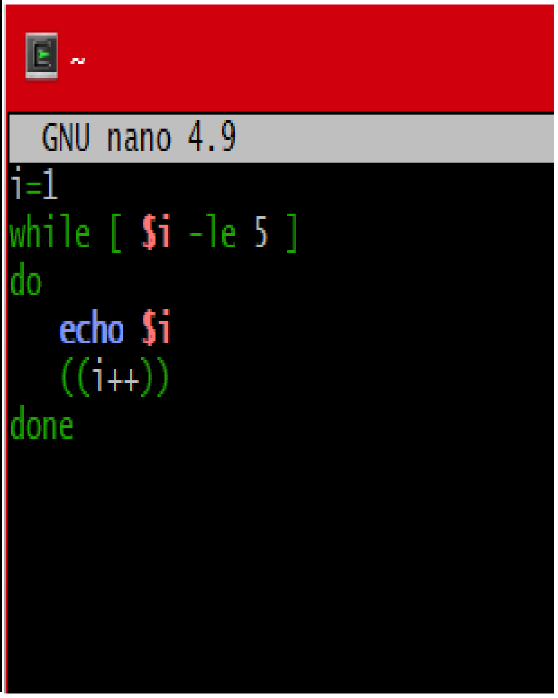


```
~
GNU nano 4.9
for i in {1..5}
do
   echo $i
done
```

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

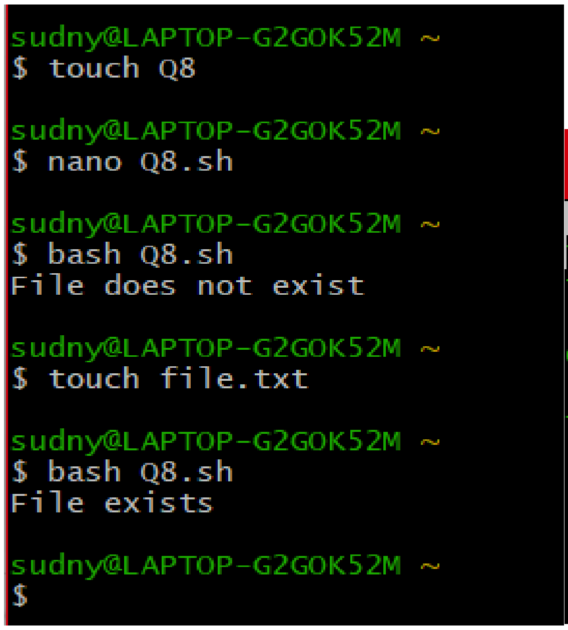


```
sudny@LAPTOP-G2GOK52M ~
$ touch Q7

sudny@LAPTOP-G2GOK52M ~
$ nano Q7.sh

sudny@LAPTOP-G2GOK52M ~
$ bash Q7.sh
Q7.sh: line 2: [: missing `]'

sudny@LAPTOP-G2GOK52M ~
$ nano Q7.sh

sudny@LAPTOP-G2GOK52M ~
$ bash Q7.sh
1
2
3
4
5

sudny@LAPTOP-G2GOK52M ~
$
```

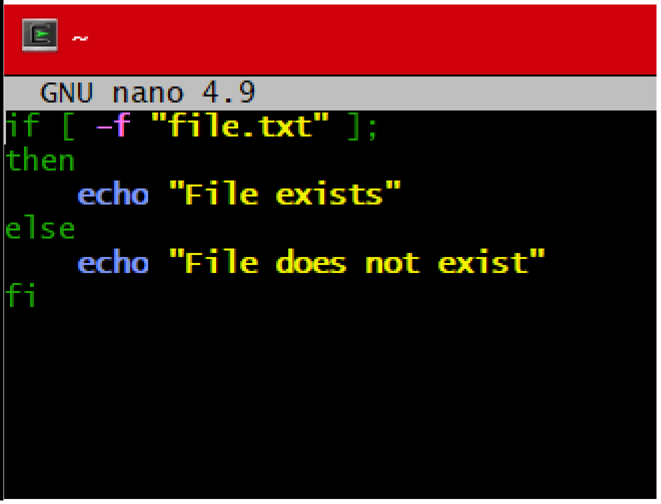


```
~
GNU nano 4.9
i=1
while [ $i -le 5 ]
do
   echo $i
   ((i++))
done
```

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".

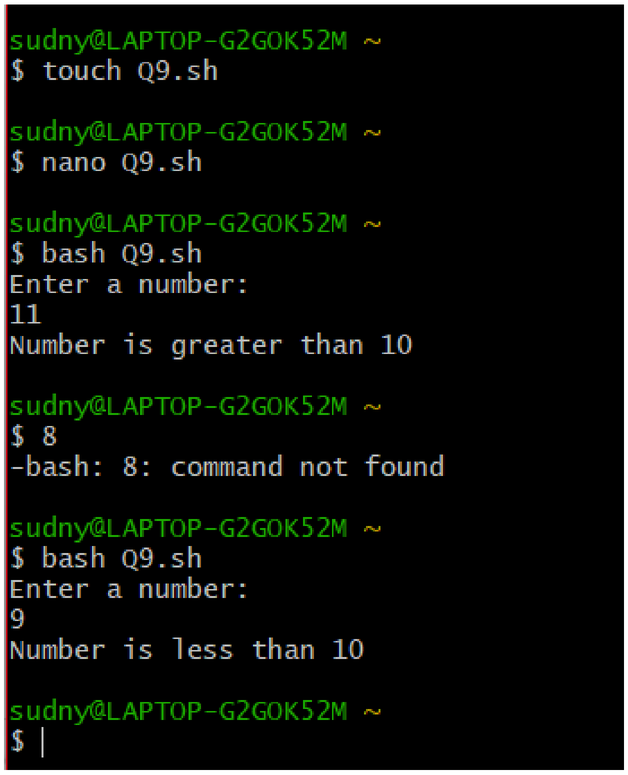


```
sudny@LAPTOP-G2GOK52M ~
$ touch Q8

sudny@LAPTOP-G2GOK52M ~
$ nano Q8.sh

sudny@LAPTOP-G2GOK52M ~
$ bash Q8.sh
File does not exist

sudny@LAPTOP-G2GOK52M ~
$ touch file.txt

sudny@LAPTOP-G2GOK52M ~
$ bash Q8.sh
File exists

sudny@LAPTOP-G2GOK52M ~
$
```

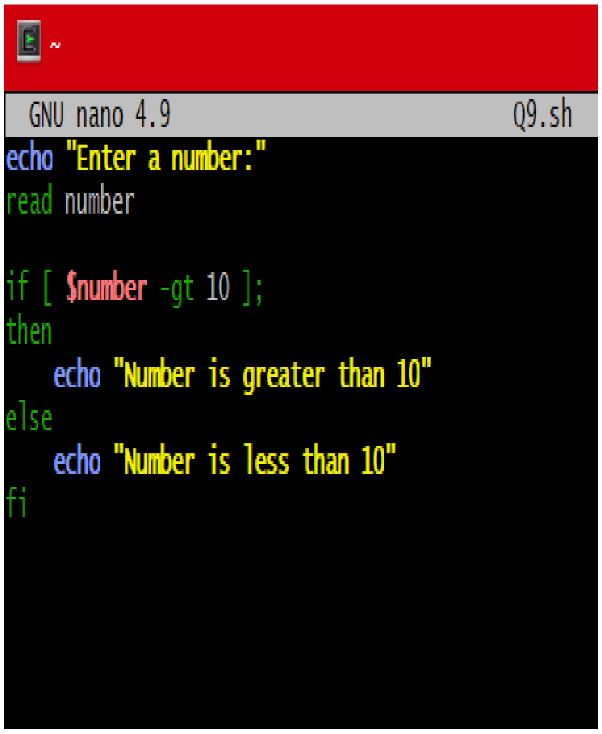


```
~
  GNU nano 4.9
if [ -f "file.txt" ];
then
    echo "File exists"
else
    echo "File does not exist"
fi
```

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.

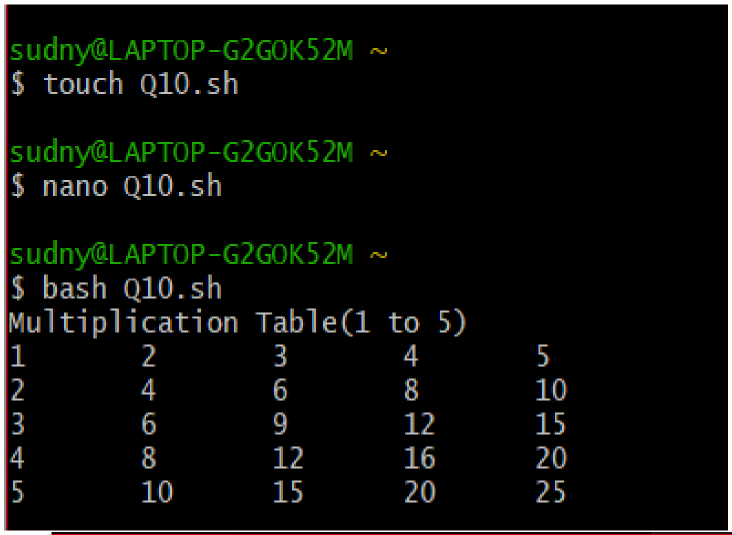


```
sudny@LAPTOP-G2GOK52M ~
$ touch Q9.sh

sudny@LAPTOP-G2GOK52M ~
$ nano Q9.sh

sudny@LAPTOP-G2GOK52M ~
$ bash Q9.sh
Enter a number:
11
Number is greater than 10

sudny@LAPTOP-G2GOK52M ~
$ 8
-bash: 8: command not found

sudny@LAPTOP-G2GOK52M ~
$ bash Q9.sh
Enter a number:
9
Number is less than 10

sudny@LAPTOP-G2GOK52M ~
$ |
```

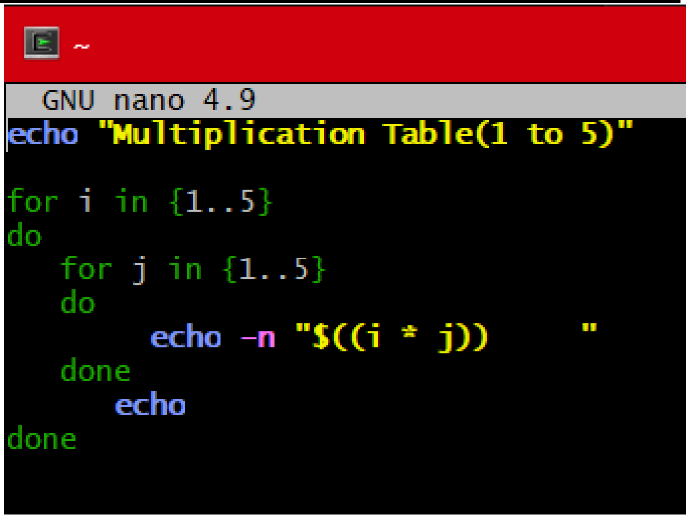


```
~
 GNU nano 4.9                                Q9.sh
echo "Enter a number:"
read number

if [ $number -gt 10 ];
then
    echo "Number is greater than 10"
else
    echo "Number is less 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.

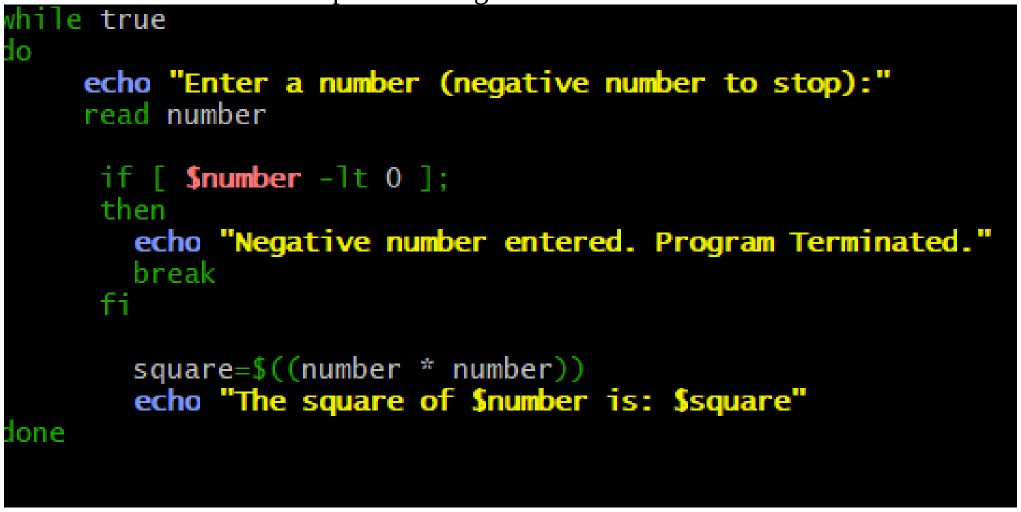


```
sudny@LAPTOP-G2GOK52M ~
$ touch Q10.sh

sudny@LAPTOP-G2GOK52M ~
$ nano Q10.sh

sudny@LAPTOP-G2GOK52M ~
$ bash Q10.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
```

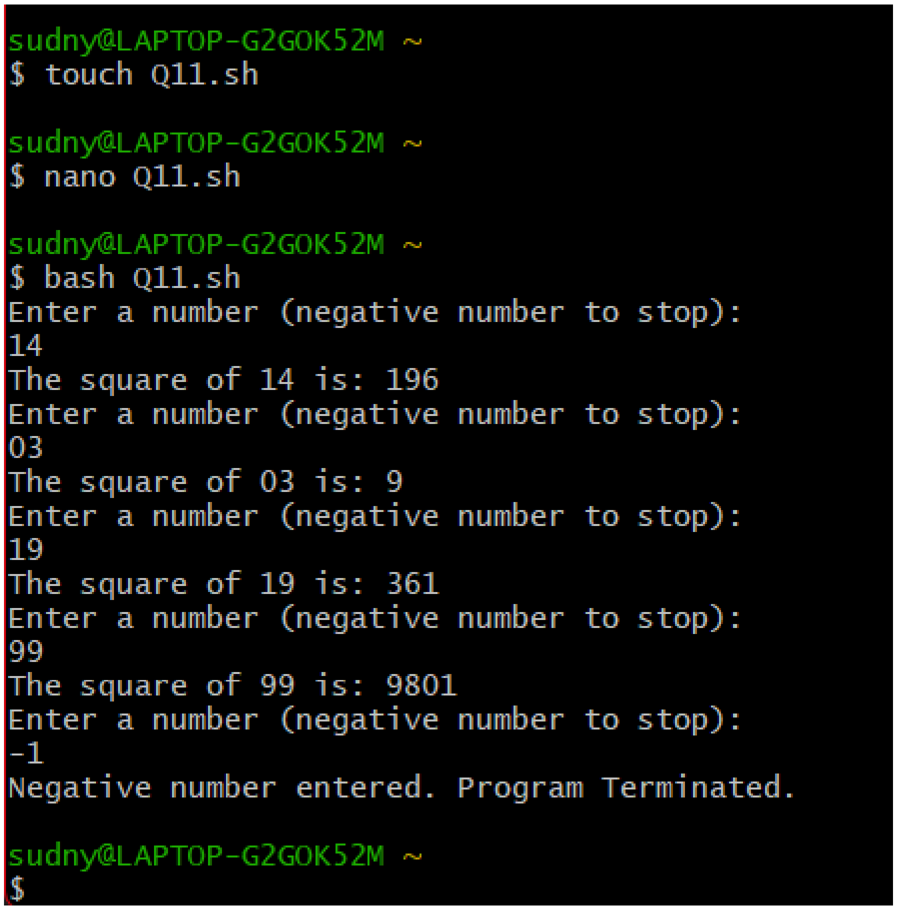


```
  GNU nano 4.9
echo "Multiplication Table(1 to 5)"

for i in {1..5}
do
   for j in {1..5}
   do
        echo -n "$((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



```
while true
do
     echo "Enter a number (negative number to stop):"
     read number

      if [ $number -lt 0 ];
      then
        echo "Negative number entered. Program Terminated."
        break
      fi

        square=$((number * number))
        echo "The square of $number is: $square"
done
```



```
sudny@LAPTOP-G2GOK52M ~
$ touch Q11.sh

sudny@LAPTOP-G2GOK52M ~
$ nano Q11.sh

sudny@LAPTOP-G2GOK52M ~
$ bash Q11.sh
Enter a number (negative number to stop):
14
The square of 14 is: 196
Enter a number (negative number to stop):
03
The square of 03 is: 9
Enter a number (negative number to stop):
19
The square of 19 is: 361
Enter a number (negative number to stop):
99
The square of 99 is: 9801
Enter a number (negative number to stop):
-1
Negative number entered. Program Terminated.

sudny@LAPTOP-G2GOK52M ~
$
```

.

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.

SOLUTION - Waiting time for processes

P1: It starts executing at time 0, so the waiting time is 0.
P2: P2 arrives at time 1 and finishes at time 5. So waiting time is 5−1= 4.
P3: P3 arrives at time 2 and finishes at time 8. So, P3's waiting time is 8−2=6.

Average time is (0+4+6)/3 = **3.33**

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.

```
SOLUTION - | P1 | P3 | P4 | P2 |      Gantt chart
             0    3   4    8   13
```

	Arrival Time	Burst Time	**Waiting Time**	**Turnaround Time**
P1	**0**	**3**	**0**	**3**
P2	**1**	**5**	**7**	**12**
P3	**2**	**1**	**1**	**2**
P4	**3**	**4**	**1**	**5**

Average Turnaround Time = (3 + 12+ 2 +5)/ 4 = 5.5

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.

SOLUTION - Gantt chart

| P1 | P2 | P4 | P3 |

0 6 10 12 19

Waiting for process - P1 : 0, P2 : 6 - 1 = 5, P3 : 12 - 2 = 10, P4 : 10 - 3 = 7

Average Waiting Time : (0+5+7+10)/4 = **5.5**

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.

SOLUTION -

| P1 | P2 | P3 | P4 | P1 | P2 | P4 | P2 |

0 2 4 6 8 10 12 13 14

	Arrival Time	Burst Time	Completion Time	TAT
P1	**0**	**4**	**10**	**10-0=10**
P2	**1**	**5**	**14**	**14-1 = 13**
P3	**2**	**2**	**6**	**6 - 2 =4**
P4	**3**	**3**	**13**	**13 - 3 = 10**

Average TAT = (10 +13+4+10)/4= 9.25

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?
   SOLUTION - After forking: The parent process increments x by 1. So, x = 6 in the parent process
   The child process increments x by 1. So, x = 6 in the child process.The final values of x are:

   Parent process: x = 6

   Child process: x = 6