Name: PIYUSH PATIL

Assignment 2 –COS

Part: A

What will the following commands do?

1) echo "Hello, World!"
   - It will print “Hello, World” after compiling.
2) name="Productive"
   - It will store string “productive” to a variable name.
3) touch file.txt
   - It will create a folder named file.txt
4) ls –a
   - It will list all the files and directories, including hidden ones.
5) rm file.txt
   - It will remove the file named file.txt
6) cp file1.txt file2.txt
   - It will copy the content of file1.txt to file2.txt
7) mv file.txt /path/to/directory/
   - It will move the file named file.txt to required directory.
8) chmod 755 script.sh
   - It will change the mode of file script.sh to Read, Wright and Execute to owner, Read and Execute to group and Read and Execute to other users.
9) grep "pattern" file.txt
   - It will find word “pattern” in file named file.txt and prints the matching lines.
10) kill PID
    - It will kill the process with an ID.
11) mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt
    - First it will make new directory named mydir
    - Changes the working directory to mydir
    - Forms a new file named file.txt
    - It will then redirect “Hello, World” to file.txt
    - Then it shows the content of file.txt using command cat
    - Terminal will show “Hello, world”
12) ls -l | grep ".txt"
    - It will list all the files in the current directory having .txt with all the details of read, write and execute.
13) cat file1.txt file2.txt | sort | uniq
    - It the show sorted and unique content (removes duplicate) of file1.txt and file2.txt
14) ls -l | grep "^d"
    - It will show only directories of current directory with details.
15) grep -r "pattern" /path/to/directory/
    - It will search for "pattern" recursively in all files within /path/to/directory/, means including all sub directories.

16) cat file1.txt file2.txt | sort | uniq –d
   - It will show only duplicate line from both the files named file2.txt and file2.txt.

17) chmod 644 file.txt
   - It will change the mode of file named file.txt to Read and Wright to owner, Read to group and Read to other users.

18) cp -r source_directory destination_directory
   - Copies source directory and all its contents recursively (including all sub directories) into destination directory.

19) find /path/to/search -name "*.txt"
   - Finds all the files with .txt

20) chmod u+x file.txt
   - Add execute mode to user in file.txt

21) echo $PATH
   - Prints the value assigned to PATH.

Part: B

Identify True or False:

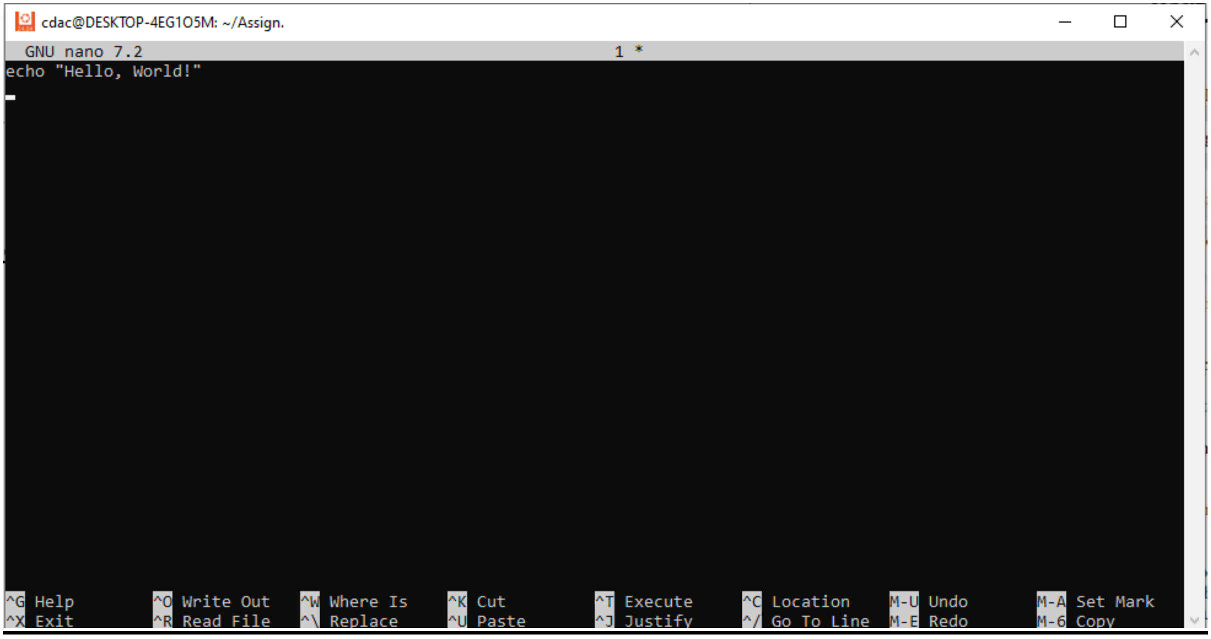
1) ls is used to list files and directories in a directory. – **TRUE**
2) mv is used to move files and directories. – **TRUE**
3) cd is used to copy files and directories. – **FALSE**, it changes directories.
4) pwd stands for "print working directory" and displays the current directory – **FALSE**, it stand for Present Working Directory.
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. – **INCORRECT**, chmod is used.
2) cpy is used to copy files and directories. – **INCORRECT**, cp is used.
3) mkfile is used to create a new file. – **INCORRECT**, touch is used.
4) catx is used to concatenate files. – **INCORRECT**, cat is used.
5) rn is used to rename files. – **INCORRECT**, rm is used.

Part: C

1) Write a shell script that prints "Hello, World!" to the terminal

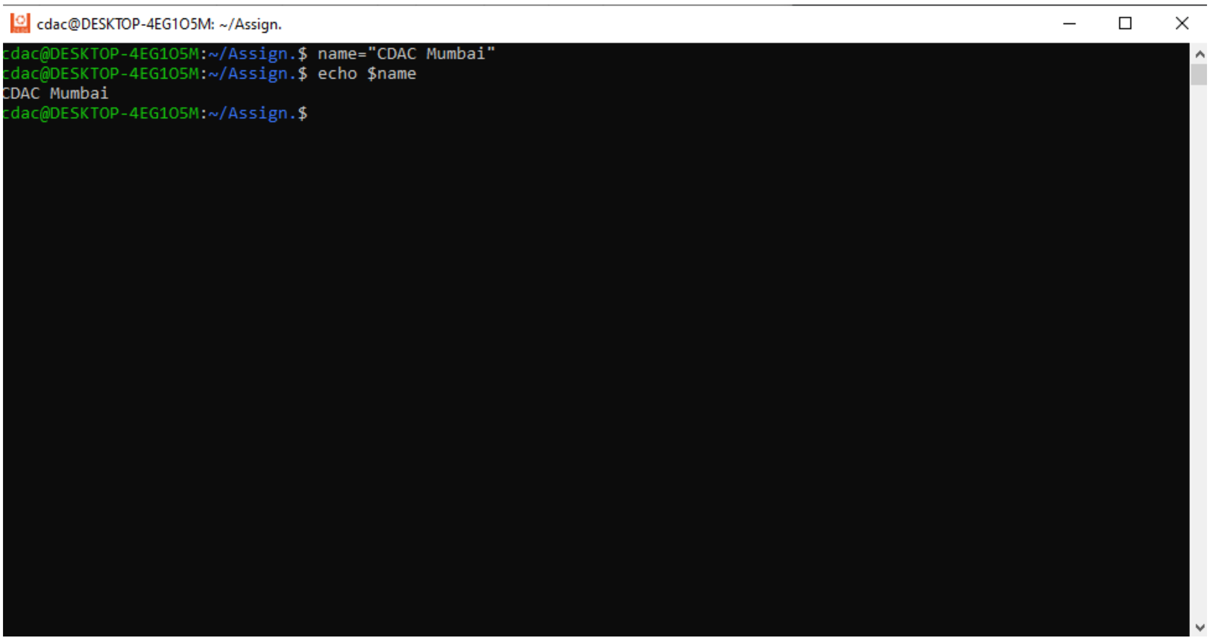


cdac@DESKTOP-4EG1O5M: ~/Assign.

```
  GNU nano 7.2                                              1 *
echo "Hello, World!"

^G Help          ^O Write Out     ^W Where Is      ^K Cut           ^T Execute       ^C Location      M-U Undo         M-A Set Mark
^X Exit          ^R Read File     ^\ Replace       ^U Paste         ^J Justify       ^/ Go To Line    M-E Redo         M-6 Copy
```

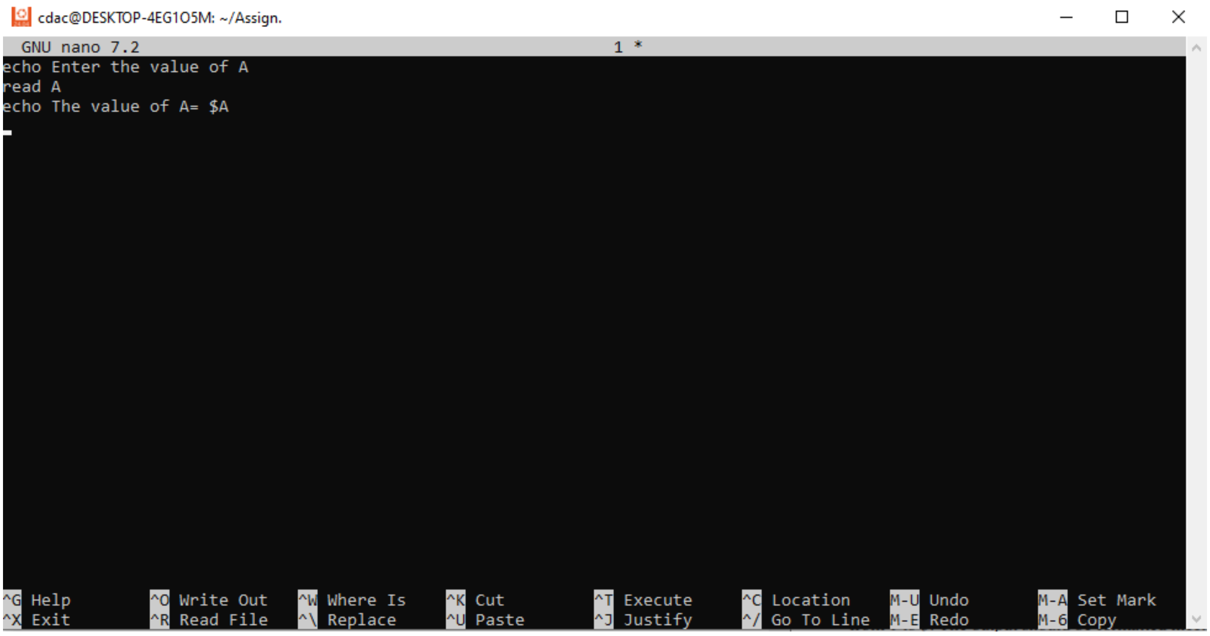
2) Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable.



cdac@DESKTOP-4EG1O5M: ~/Assign.

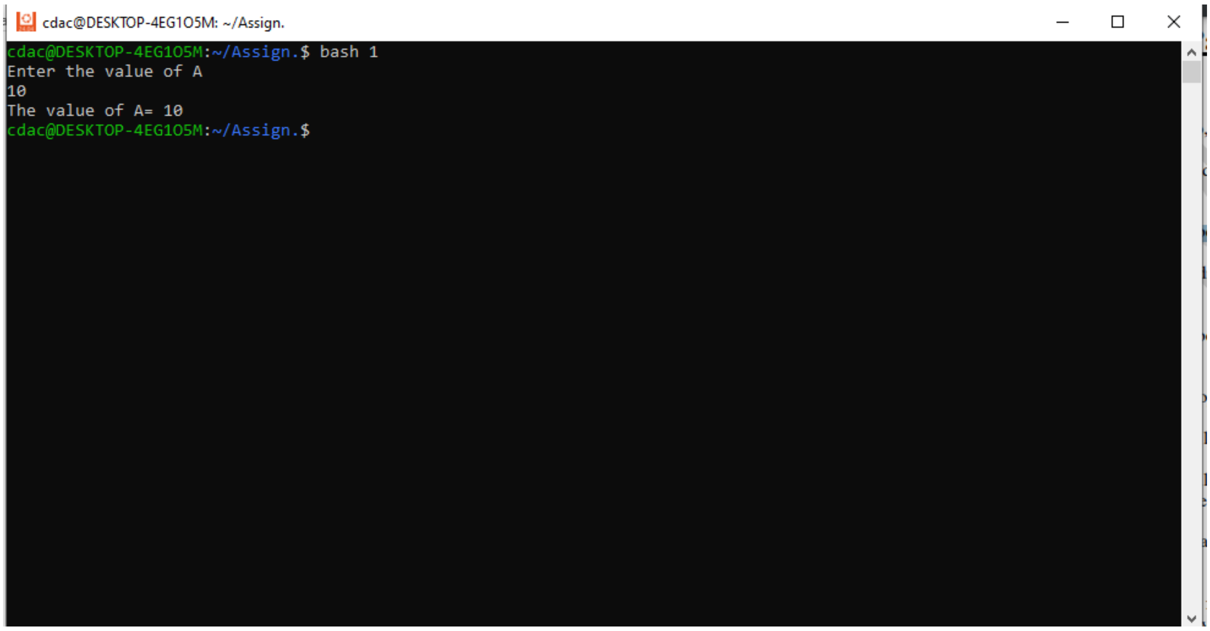
```
cdac@DESKTOP-4EG105M:~/Assign.$ name="CDAC Mumbai"
cdac@DESKTOP-4EG105M:~/Assign.$ echo $name
CDAC Mumbai
cdac@DESKTOP-4EG105M:~/Assign.$
```

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



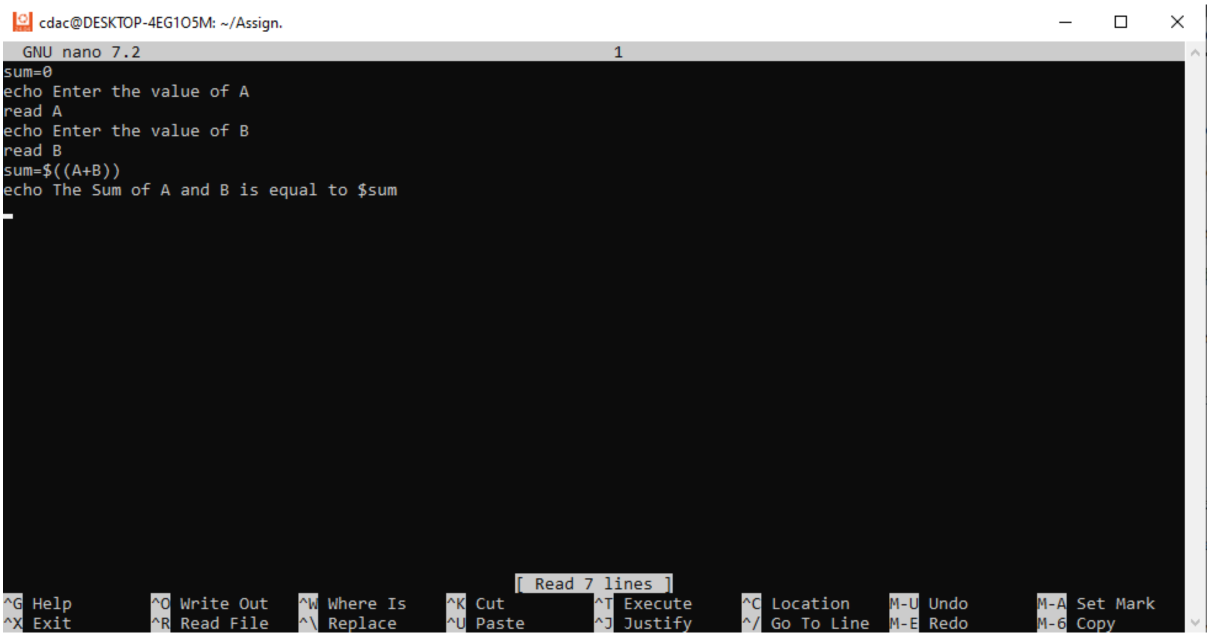
```
  GNU nano 7.2                                        1 *
echo Enter the value of A
read A
echo The value of A= $A

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy
```



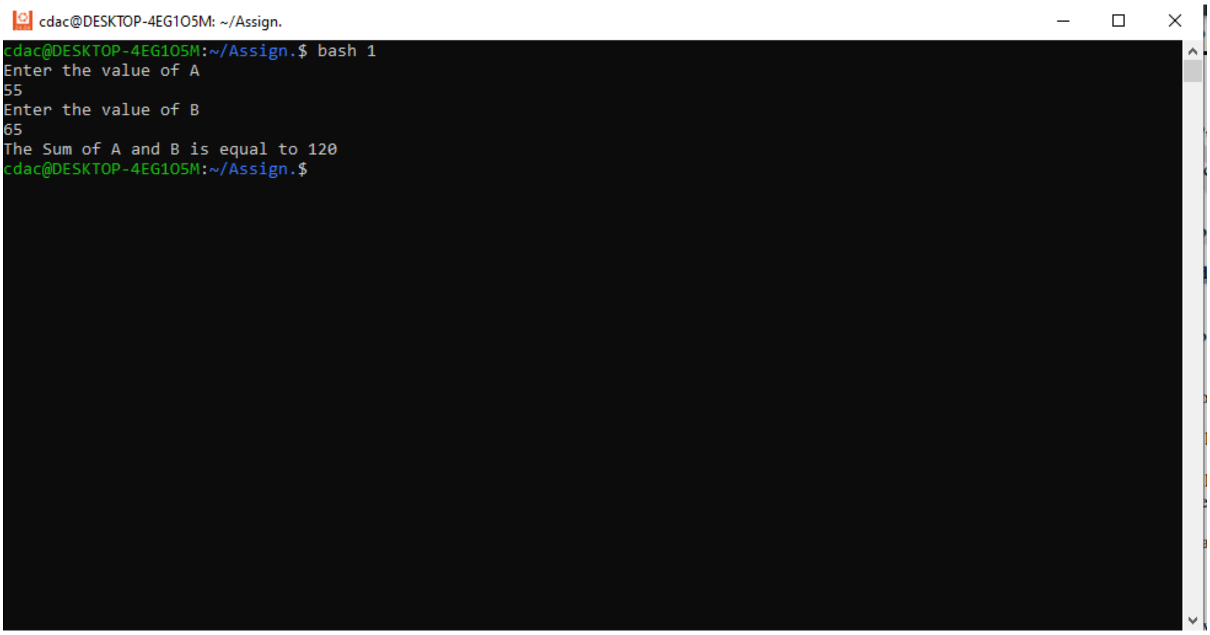
```
cdac@DESKTOP-4EG105M:~/Assign.$ bash 1
Enter the value of A
10
The value of A= 10
cdac@DESKTOP-4EG105M:~/Assign.$
```

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



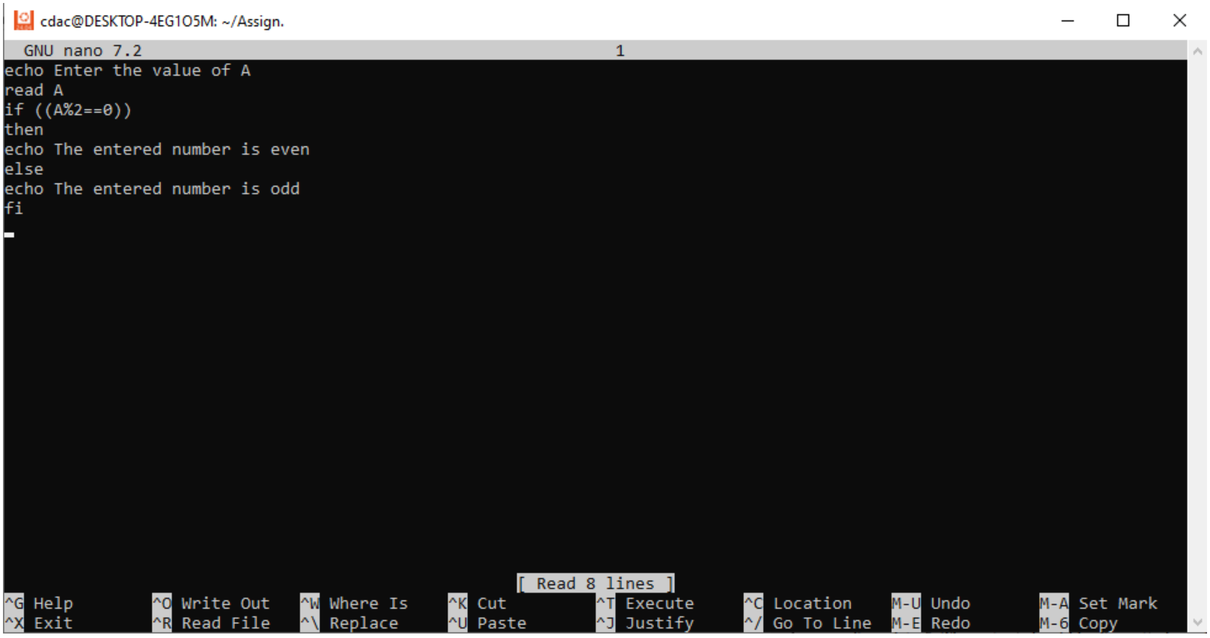
```
  GNU nano 7.2                                           1
sum=0
echo Enter the value of A
read A
echo Enter the value of B
read B
sum=$((A+B))
echo The Sum of A and B is equal to $sum

                                                    [ Read 7 lines ]
^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy
```



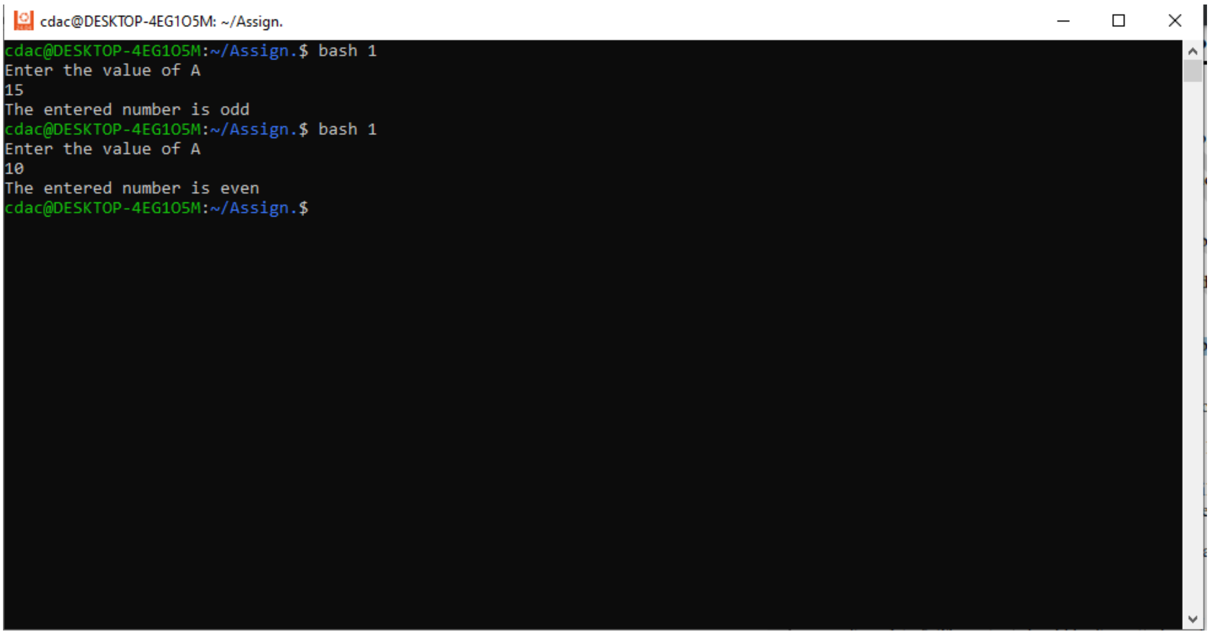
```
cdac@DESKTOP-4EG1O5M:~/Assign.$ bash 1
Enter the value of A
55
Enter the value of B
65
The Sum of A and B is equal to 120
cdac@DESKTOP-4EG1O5M:~/Assign.$
```

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



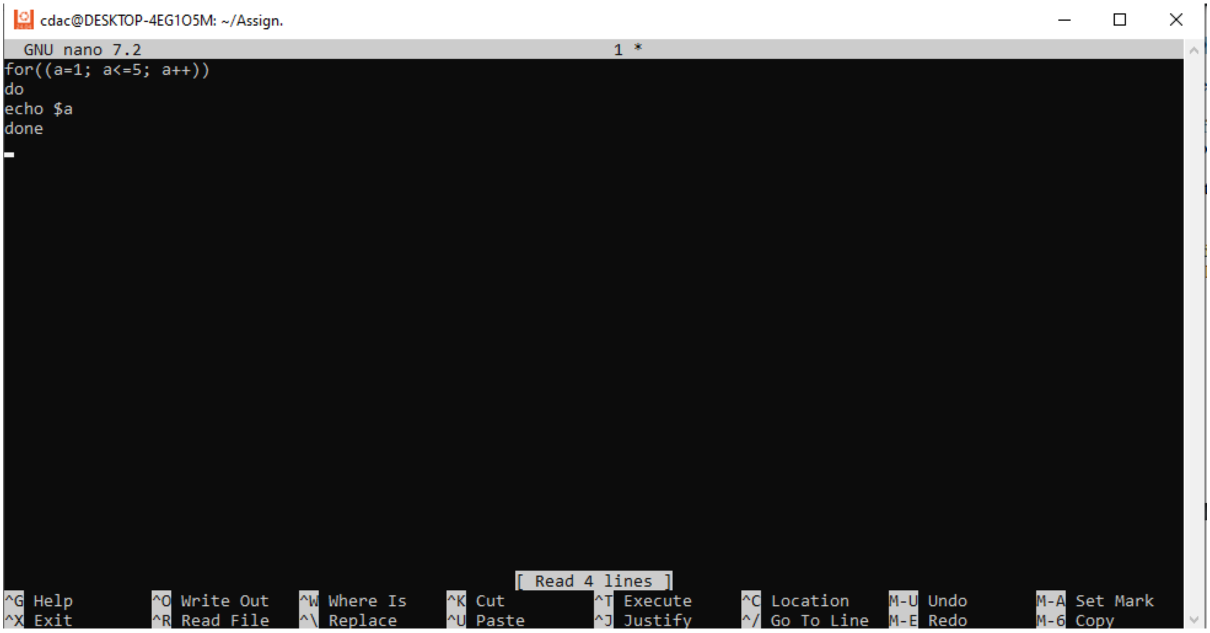
```
  GNU nano 7.2                                          1
echo Enter the value of A
read A
if ((A%2==0))
then
echo The entered number is even
else
echo The entered number is odd
fi

                                    [ Read 8 lines ]
^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-6 Copy
```



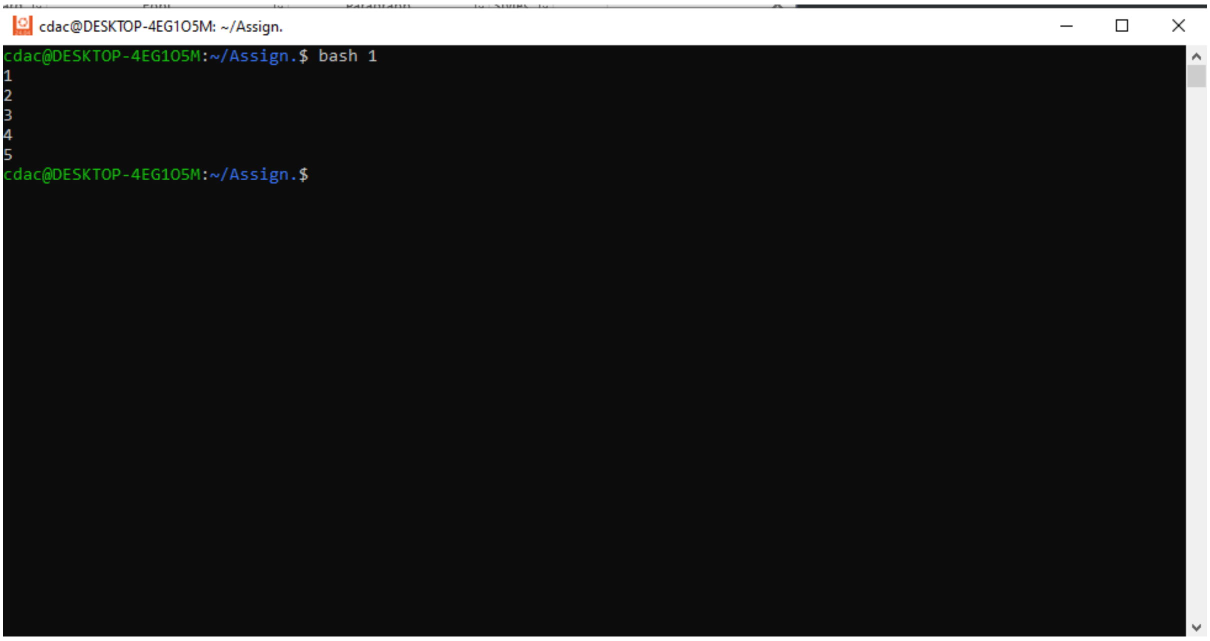
```
cdac@DESKTOP-4EG1O5M:~/Assign.$ bash 1
Enter the value of A
15
The entered number is odd
cdac@DESKTOP-4EG1O5M:~/Assign.$ bash 1
Enter the value of A
10
The entered number is even
cdac@DESKTOP-4EG1O5M:~/Assign.$
```

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



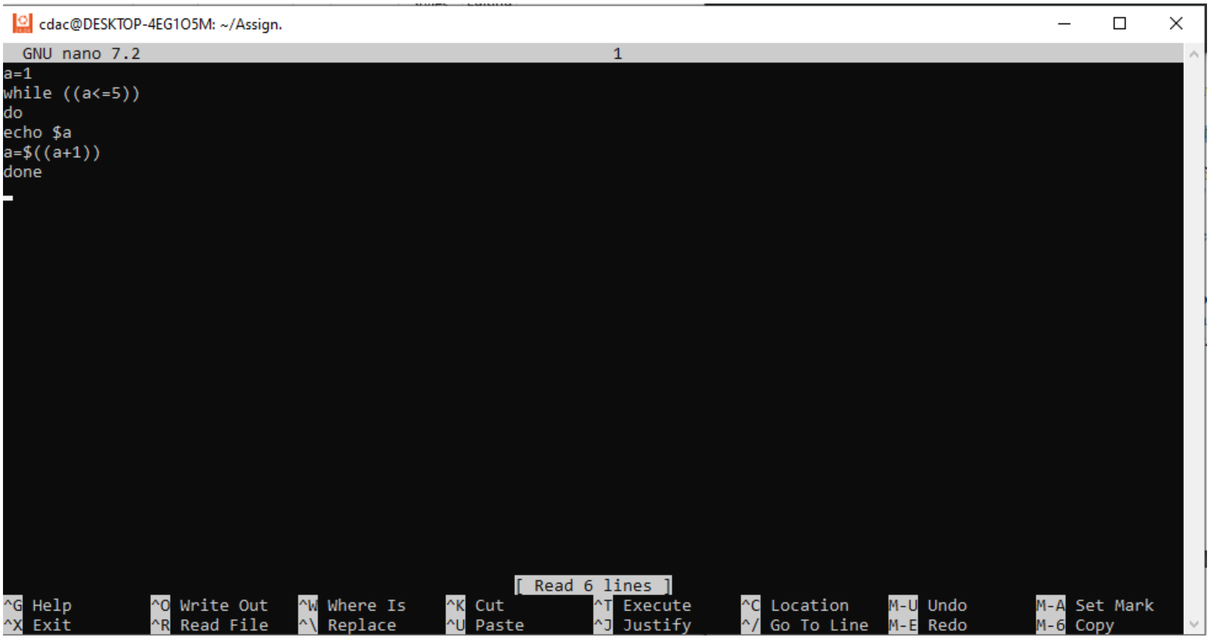
```
cdac@DESKTOP-4EG1O5M: ~/Assign.
 GNU nano 7.2                                         1 *
for((a=1; a<=5; a++))
do
echo $a
done

                                                [ Read 4 lines ]
^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy
```



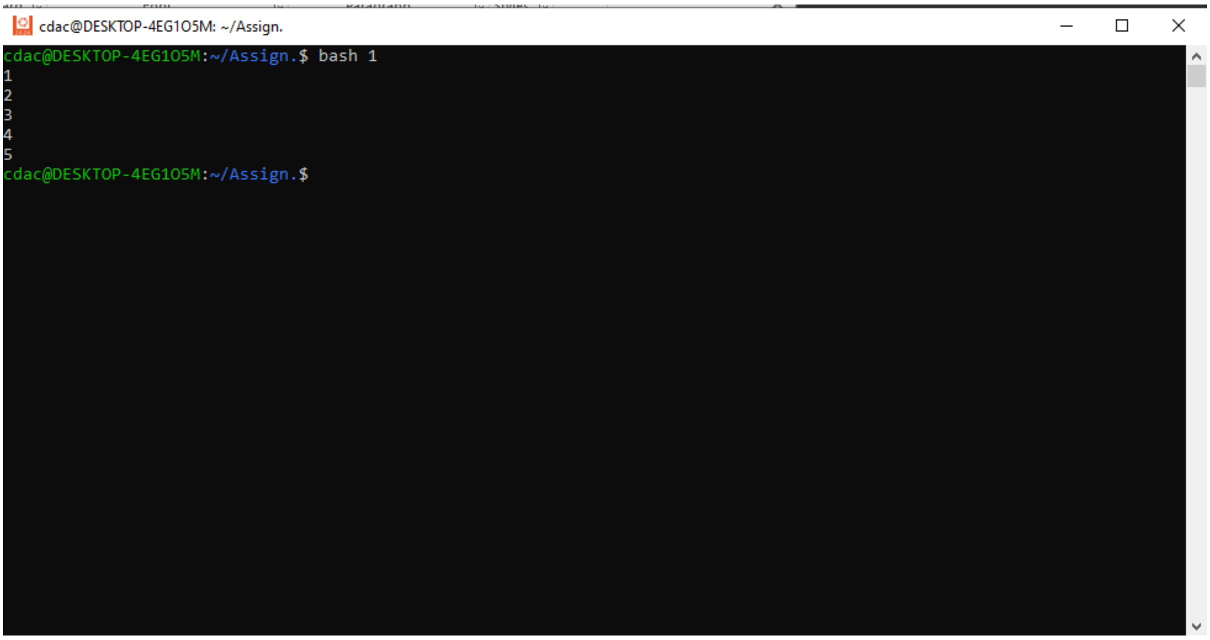
```
cdac@DESKTOP-4EG1O5M: ~/Assign.
cdac@DESKTOP-4EG105M:~/Assign.$ bash 1
1
2
3
4
5
cdac@DESKTOP-4EG105M:~/Assign.$
```

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



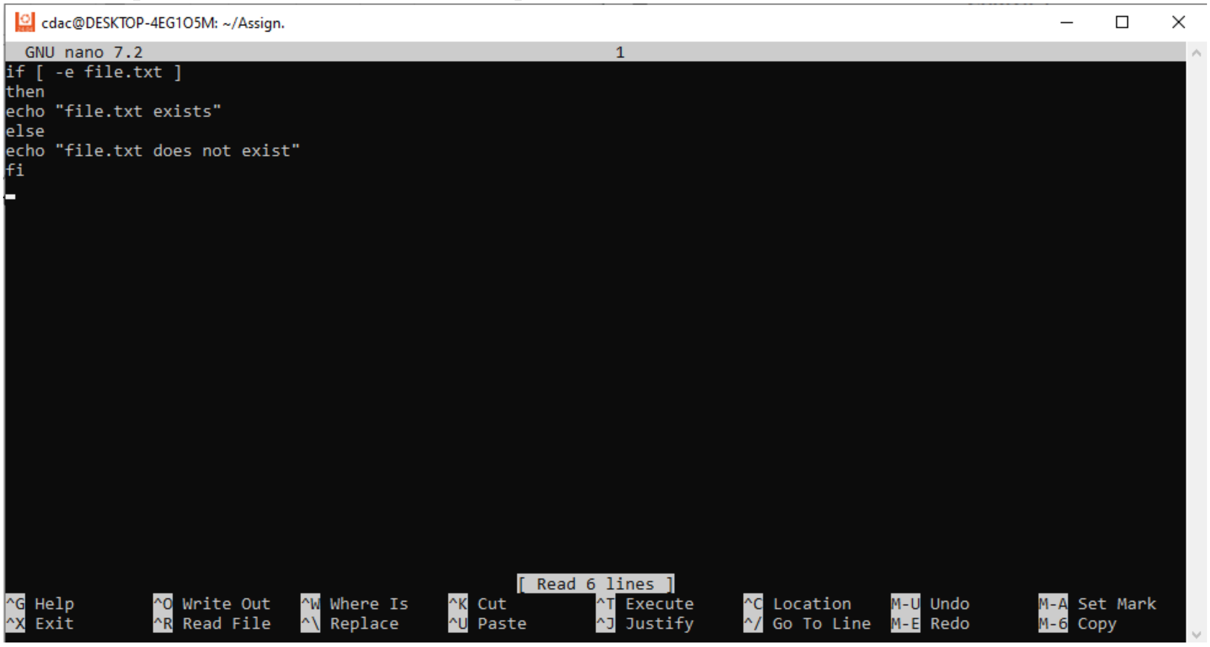
```
  GNU nano 7.2                                  1
a=1
while ((a<=5))
do
echo $a
a=$((a+1))
done

                                          [ Read 6 lines ]
^G Help          ^O Write Out     ^W Where Is      ^K Cut           ^T Execute       ^C Location      M-U Undo         M-A Set Mark
^X Exit          ^R Read File     ^\ Replace       ^U Paste         ^J Justify       ^/ Go To Line    M-E Redo         M-6 Copy
```



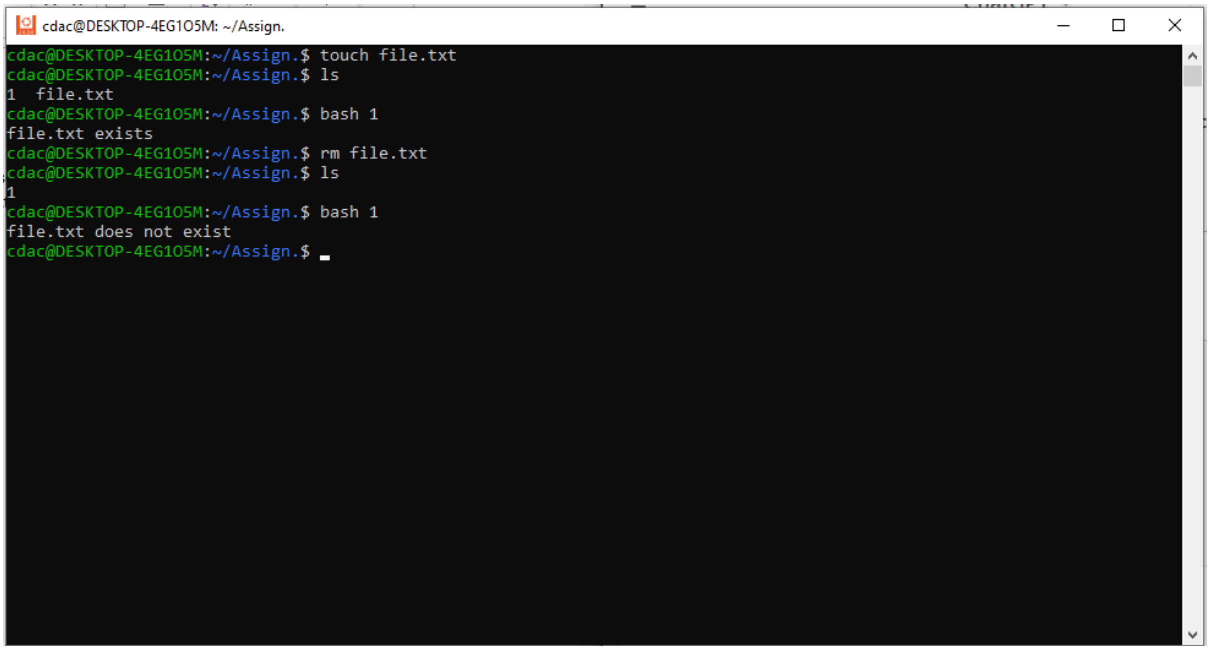
```
cdac@DESKTOP-4EG1O5M:~/Assign.$ bash 1
1
2
3
4
5
cdac@DESKTOP-4EG1O5M:~/Assign.$
```

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".



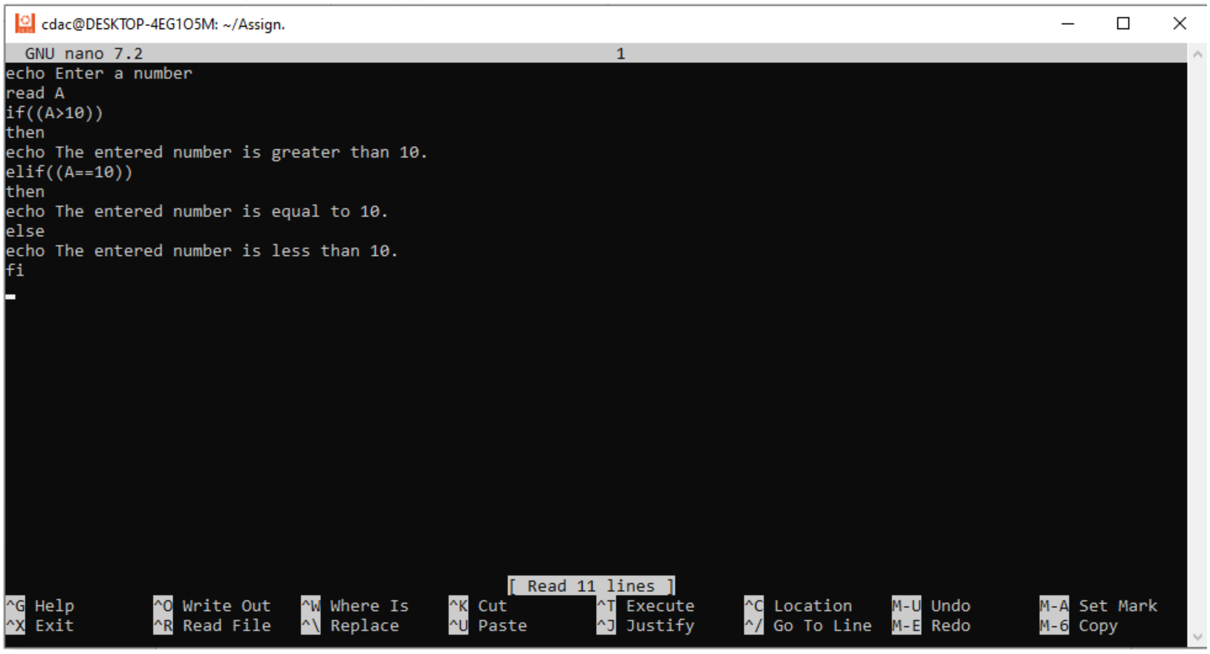
```
  GNU nano 7.2                                            1
if [ -e file.txt ]
then
echo "file.txt exists"
else
echo "file.txt does not exist"
fi

                                                    [ Read 6 lines ]
^G Help         ^O Write Out    ^W Where Is     ^K Cut          ^T Execute      ^C Location     M-U Undo        M-A Set Mark
^X Exit         ^R Read File    ^\ Replace      ^U Paste        ^J Justify      ^/ Go To Line   M-E Redo        M-6 Copy
```



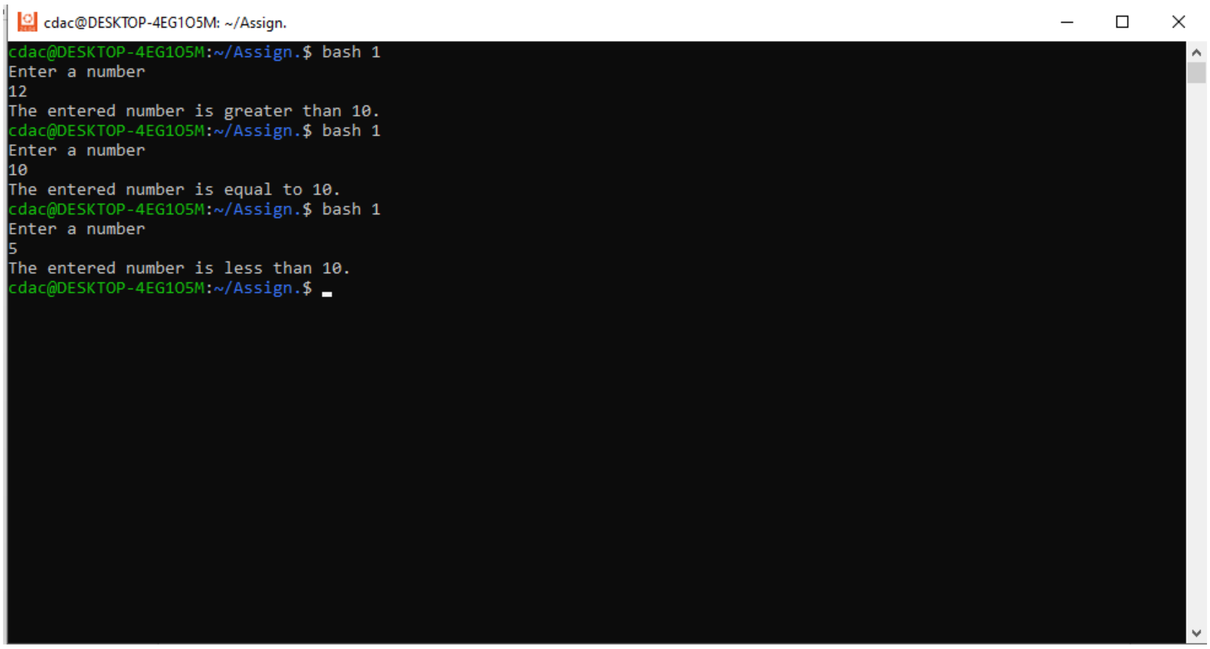
```
cdac@DESKTOP-4EG105M:~/Assign.$ touch file.txt
cdac@DESKTOP-4EG105M:~/Assign.$ ls
1  file.txt
cdac@DESKTOP-4EG105M:~/Assign.$ bash 1
file.txt exists
cdac@DESKTOP-4EG105M:~/Assign.$ rm file.txt
cdac@DESKTOP-4EG105M:~/Assign.$ ls
1
cdac@DESKTOP-4EG105M:~/Assign.$ bash 1
file.txt does not exist
cdac@DESKTOP-4EG105M:~/Assign.$
```

9) Write a shell script that uses the if statement to check if a number is greater than 10 and prints a message accordingly.



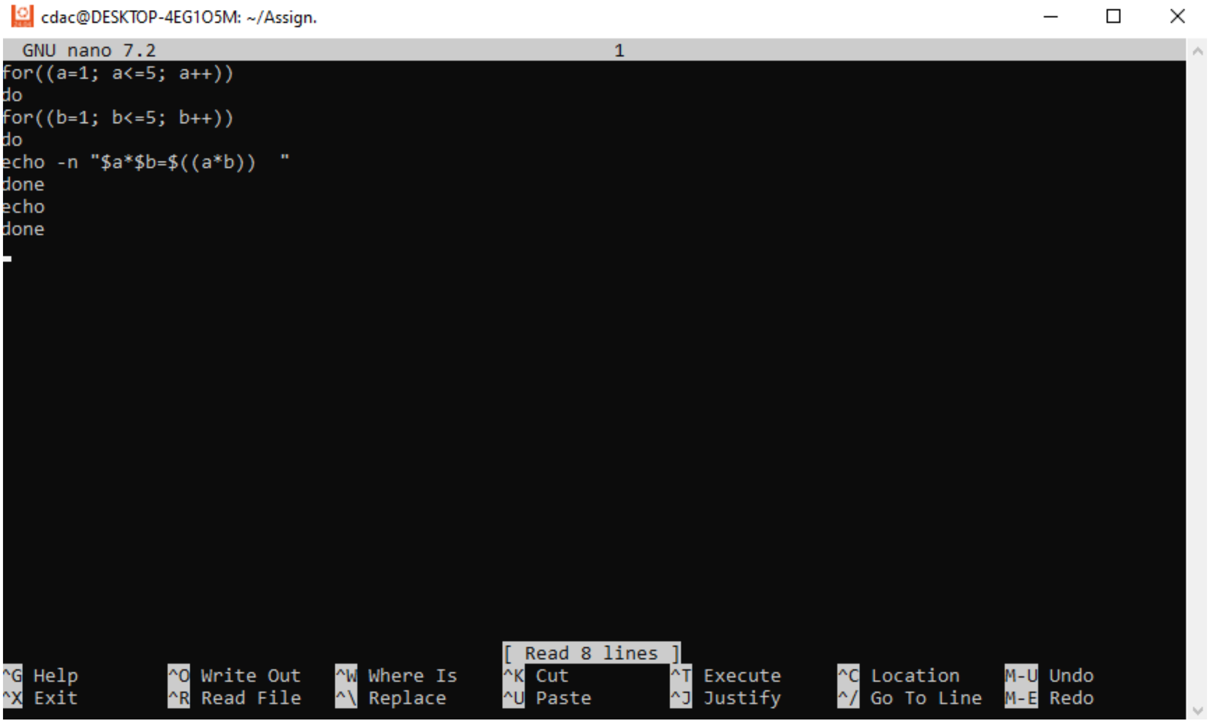
```
  GNU nano 7.2                                              1
echo Enter a number
read A
if((A>10))
then
echo The entered number is greater than 10.
elif((A==10))
then
echo The entered number is equal to 10.
else
echo The entered number is less than 10.
fi

                                                   [ Read 11 lines ]
^G Help         ^O Write Out    ^W Where Is     ^K Cut          ^T Execute      ^C Location     M-U Undo        M-A Set Mark
^X Exit         ^R Read File    ^\ Replace      ^U Paste        ^J Justify      ^/ Go To Line   M-E Redo        M-6 Copy
```

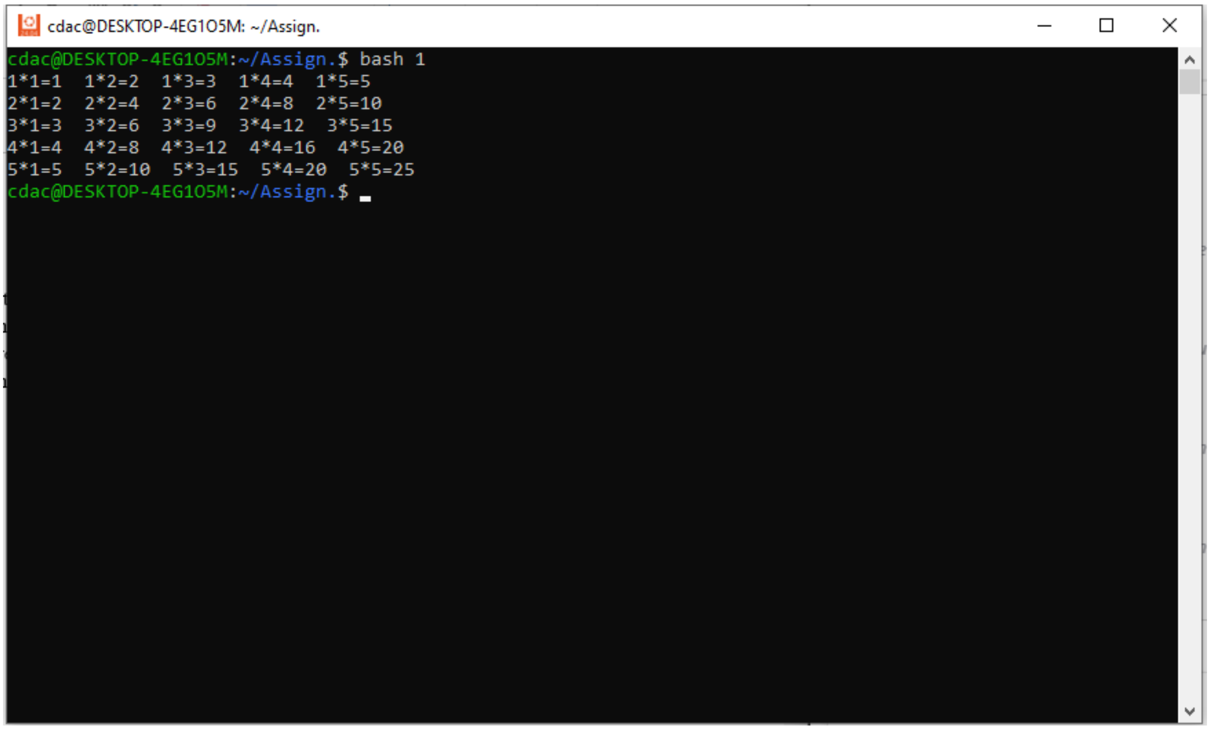


```
cdac@DESKTOP-4EG105M:~/Assign.$ bash 1
Enter a number
12
The entered number is greater than 10.
cdac@DESKTOP-4EG105M:~/Assign.$ bash 1
Enter a number
10
The entered number is equal to 10.
cdac@DESKTOP-4EG105M:~/Assign.$ bash 1
Enter a number
5
The entered number is less than 10.
cdac@DESKTOP-4EG105M:~/Assign.$ 
```

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.

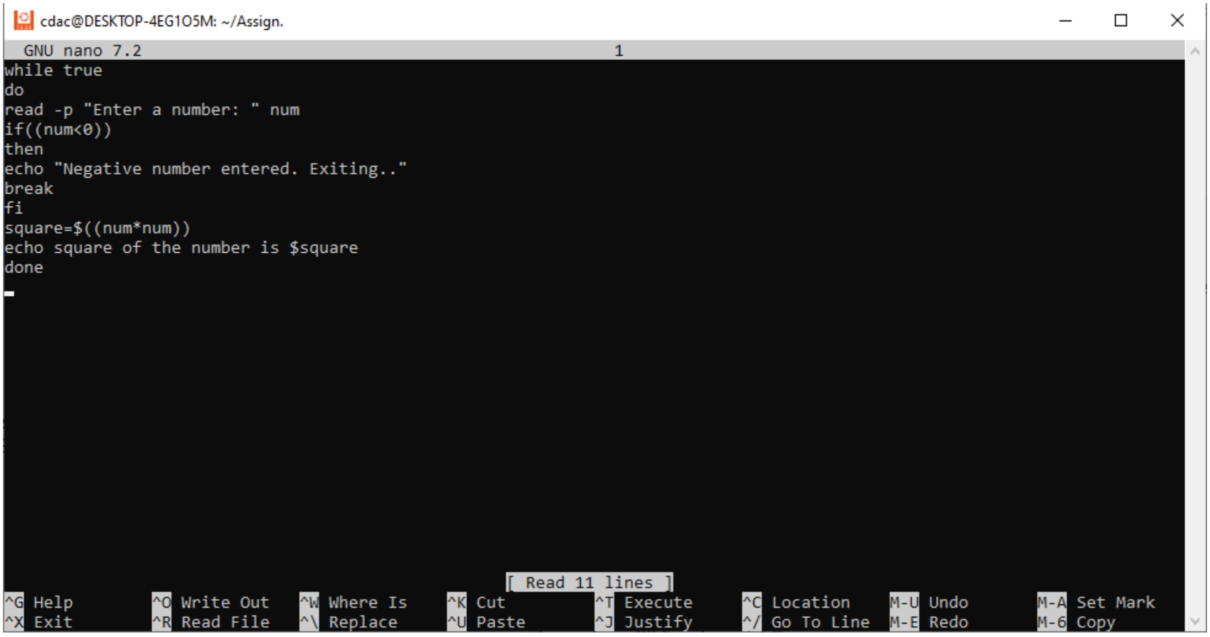


```
for((a=1; a<=5; a++))
do
for((b=1; b<=5; b++))
do
echo -n "$a*$b=$((a*b))  "
done
echo
done
```



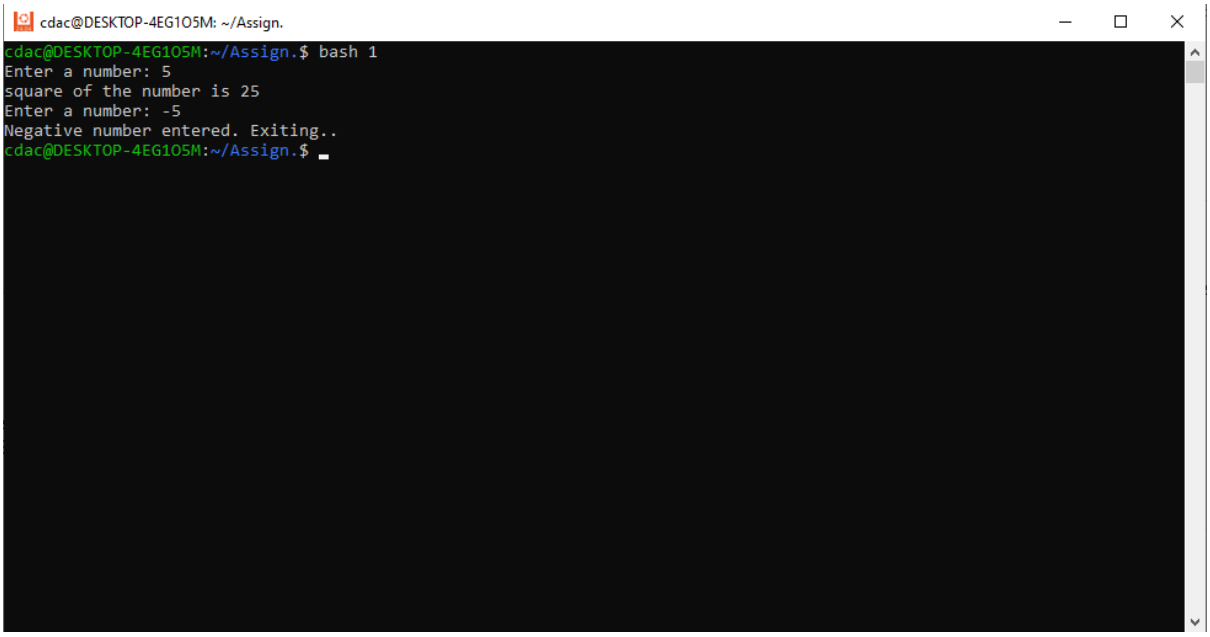
```
cdac@DESKTOP-4EG1O5M:~/Assign.$ bash 1
1*1=1  1*2=2  1*3=3  1*4=4  1*5=5
2*1=2  2*2=4  2*3=6  2*4=8  2*5=10
3*1=3  3*2=6  3*3=9  3*4=12  3*5=15
4*1=4  4*2=8  4*3=12  4*4=16  4*5=20
5*1=5  5*2=10  5*3=15  5*4=20  5*5=25
cdac@DESKTOP-4EG1O5M:~/Assign.$
```

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 7.2                                            1
while true
do
read -p "Enter a number: " num
if((num<0))
then
echo "Negative number entered. Exiting.."
break
fi
square=$((num*num))
echo square of the number is $square
done
```

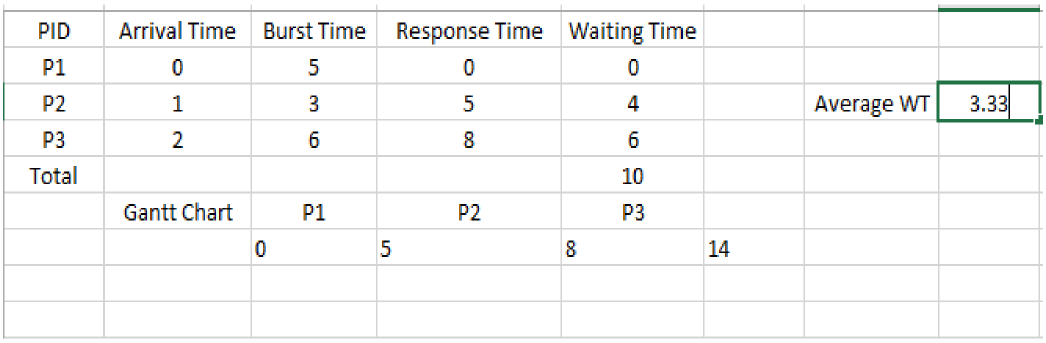
```
                                                   [ Read 11 lines ]
^G Help         ^O Write Out    ^W Where Is     ^K Cut          ^T Execute      ^C Location     M-U Undo        M-A Set Mark
^X Exit         ^R Read File    ^\ Replace      ^U Paste        ^J Justify      ^/ Go To Line   M-E Redo        M-6 Copy
```



```
cdac@DESKTOP-4EG1O5M:~/Assign.$ bash 1
Enter a number: 5
square of the number is 25
Enter a number: -5
Negative number entered. Exiting..
cdac@DESKTOP-4EG1O5M:~/Assign.$
```

Part: D

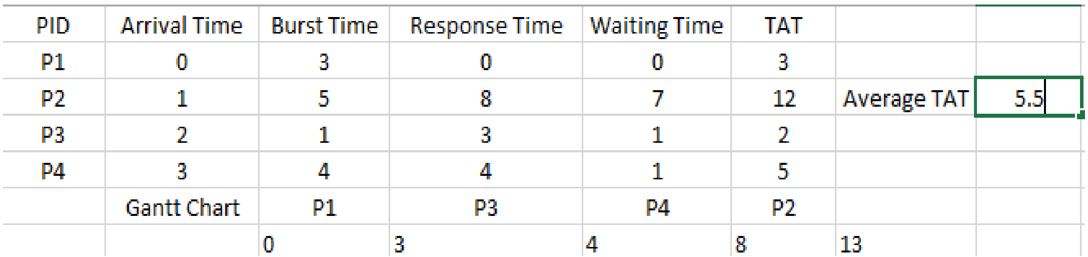
1) Consider the following processes with arrival times and burst times. Calculate the average waiting time using First-Come, First-Served (FCFS) scheduling.



PID	Arrival Time	Burst Time	Response Time	Waiting Time			
P1	0	5	0	0			
P2	1	3	5	4		Average WT	3.33
P3	2	6	8	6			
Total				10			
	Gantt Chart	P1	P2	P3			
		0	5	8	14		

Average Waiting Time = 3.33

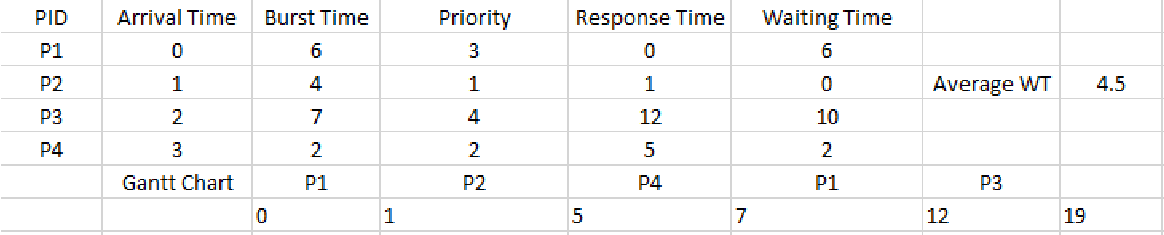
2) Consider the following processes with arrival times and burst times. Calculate the average turnaround time using Shortest Job First (SJF) scheduling.



PID	Arrival Time	Burst Time	Response Time	Waiting Time	TAT		
P1	0	3	0	0	3		
P2	1	5	8	7	12	Average TAT	5.5
P3	2	1	3	1	2		
P4	3	4	4	1	5		
	Gantt Chart	P1	P3	P4	P2		
		0	3	4	8	13	

Average Turnaround time = 5.5

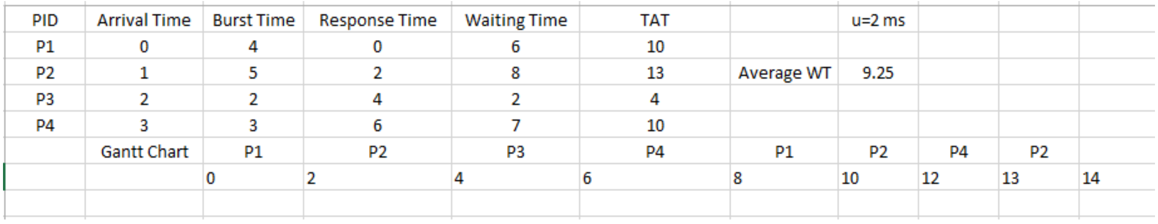
3) Consider the following processes with arrival times, burst times, and priorities (lower number indicates higher priority). Calculate the average waiting time using Priority Scheduling.



PID	Arrival Time	Burst Time	Priority	Response Time	Waiting Time		
P1	0	6	3	0	6		
P2	1	4	1	1	0	Average WT	4.5
P3	2	7	4	12	10		
P4	3	2	2	5	2		
	Gantt Chart	P1	P2	P4	P1	P3	
		0	1	5	7	12	19

Average Waiting Time = 4.5

4) Consider the following processes with arrival times and burst times, and the time quantum for Round Robin scheduling is 2 units. Calculate the average turnaround time using Round Robin scheduling.



PID	Arrival Time	Burst Time	Response Time	Waiting Time	TAT		u=2 ms			
P1	0	4	0	6	10					
P2	1	5	2	8	13	Average WT	9.25			
P3	2	2	4	2	4					
P4	3	3	6	7	10					
	Gantt Chart	P1	P2	P3	P4	P1	P2	P4	P2	
		0	2	4	6	8	10	12	13	14

Average Turnaround Time = 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?

Parent Process = 6
Child Process = 6