CDAC MUMBAI

Concepts of Operating System

Assignment 2

GOURAV SAHU_KH

PART-A :-

What will the following commands do?

• echo "Hello, World!"

Ans— This command will print “Hello World” on the terminal.

• name="Productive"

Ans--- This command will declare a variable “name” with value “Productive” so whenever we use command echo $name so it will print Productive on terminal. This process known as Shell Variable.

• touch file.txt

Ans--- This command will create a file named as “file.txt” under that particular directory in which the user is working.

• ls -a

Ans--- This command displays all the above information regarding all the files and the directories.

• rm file.txt

Ans—This is command used to delete a file from the directory, so it will delete the file named as “file.txt”.

• cp file1.txt file2.txt

Ans--- This command used to copy the content of a file to another file, In this command content of file1.txt will be copied to file2.txt. If file2.txt does not exist so it will create the file2.txt file then copy the content.

• mv file.txt /path/to/directory/

Ans--- This mv command used to move the file from one directory to another. In this command file.txt will be moved to the directory whose path and name will be given with the command.

• chmod 755 script.sh

Ans--- The chmod command used to change the permissions of the file. So this command will allow Read, Write and Execute permission of the file “script.sh” to owner and Read, Execute permission to group and others.

- **grep "pattern" file.txt**

Ans--- This command used to search a specific word in the file. This command will search for word “pattern” in the file named as file.txt.

- **kill PID**

Ans—This command used to send a signal to the process for termination with a specific Process ID. If Process ID is 1234 so the command will be kill 1234. So the process will be terminated.

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

Ans--- This the process to give multiple commands in a single line or we can say at same time. && operator used to separate the commands. Command “mkdir mydir” will create a directory name as mydir and the next command “cd mydir” will make user to enter in the directory mydir. Then command “touch file.txt” will make a file inside the directory mydir. Command echo “Hello world” will print Hello World on terminal but it will be used as input for the file named as “file.txt” so the output of echo command will be saved to file.txt because of redirection (>).Then cat command will display the content of file.txt which is Hello World.

- **ls -l | grep ".txt"**

Ans--- Command ls -l used to lists files and directories in long format, showing details like permissions, owner, size. (|) This is the symbol of piping, So the output of ls -l command will be the output of grep command then grep will search the lines with “.txt”.

- **cat file1.txt file2.txt | sort | uniq**

Ans--- cat command is used to print the file contents so it will be used to print both files file1.txt and file2.txt but the content of both files will be the input for sort command because of piping(|). sort command used to sort the content in alphabetic order again sorted content of both file will be the input for uniq command which is used to show unique value from the content so it will remove the duplicate entries and will display the unique values.

- **ls -l | grep "^d"**

Ans--- Command ls -l used to lists files and directories in long format, showing details like permissions, owner, size. Then this output will become input for grep command because of piping and the grep will display all the directories because “^d” indicates directories.

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

Ans--- grep command is used to search a specific word in any file. In grep -r command -r tells grep to search recursively in all files and subdirectories under the particular whose path given as a argument with grep -r command. “pattern” is the word the grep command will look for.

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

Ans--- In this command cat is used to display content of the file on terminal so cat will display the content of both file named as file1.txt and file2.txt then this output of cat command will be the input for sort command because of pipe (|) and sort command will sort the output of sort (content of file1.txt and file2.txt) in alphabetic order and this output of sort will become the input for uniq -d command will display only duplicate lines from the content of both files.

• **chmod 644 file.txt**

Ans--- chmod command is used to change or give the permissions to owner, groups and others of a file. There are 3 types of permissions – Read(4), Write(2), Execute(1). The numbering with permissions is a presentation of permission in numeric form. In this chmod 644 file.txt command 6(4 is for read and 2 is for write) is for owner/user it means read and write permission of "file.txt" is given to the owner. Then 4 (read) is for groups, read permission is given to the groups and same as groups, 4(read) permission is also given to the others.

• **cp -r source_directory destination_directory**

Ans--- cp command is used to copy the files and directories. -r tells cp command to copy all the directories and file inside the source directory (The directory which you want to copy) in recursive manner. The destination directory is the directory inside which you want to copy the source directory.

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

Ans--- find command is powerful command used to search a file or directory. Followed by find we have to give the path of directory inside we want to search the file or directory.
-name "*.txt" finds files with names ending with .txt inside the directory. The * is a wildcard that matches any characters.

• **chmod u+x file.txt**

Ans—chmod command used to modify the permissions of a file. in chmod u+x file.txt where 'u' denotes the user or owner and 'x' denotes execute permission so this command will add the execute permission of file named as file.txt to the user or owner.

• **echo $PATH**

Ans--- echo command used to print text or variable on the terminal. So echo $PATH command is used to display the contents of the "PATH" environment variable, which essentially shows a list of directories where the system looks for executable programs when you run a command without specifying its full path. It shows all the directories separated by colon(:).

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. == **True**
8) **rm -rf file.txt** deletes a file forcefully without confirmation. **==True**

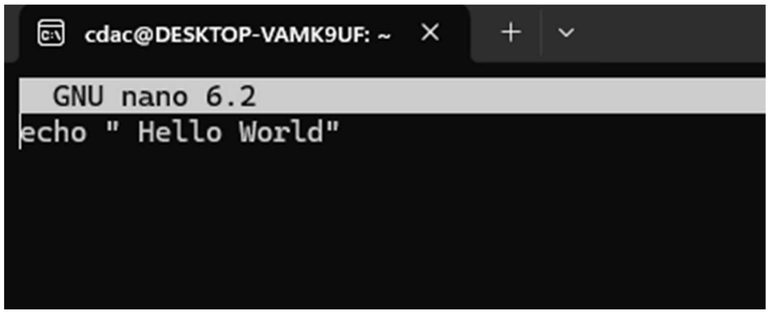
Identify the Incorrect Commands:

1) **chmodx** is used to change file permissions. **==Incorrect**
2) **cpy** is used to copy files and directories. == **Incorrect**
3) **mkfile** is used to create a new file. **== Incorrect**
4) **catx** is used to concatenate files. **== Incorrect**
5) **rn** is used to rename files. **== Incorrect**

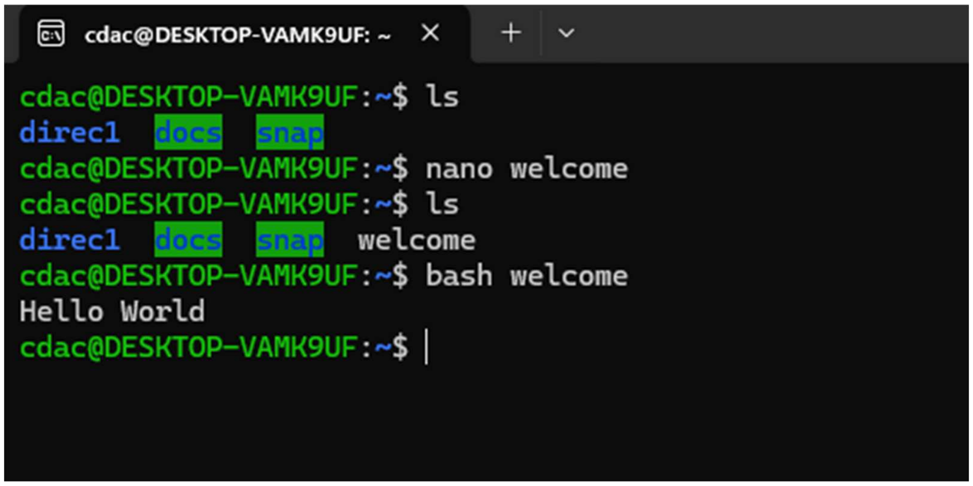
All commands are incorrect.

<u>Part C :-</u>

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

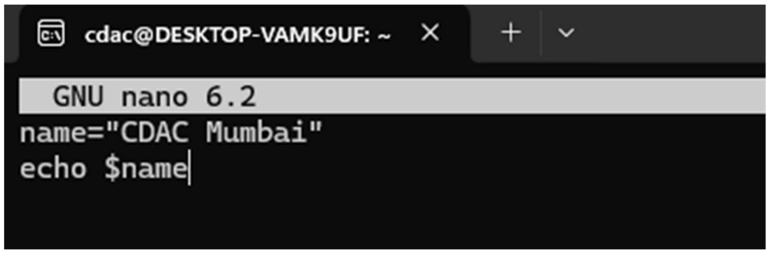


```
  GNU nano 6.2
echo " Hello World"
```

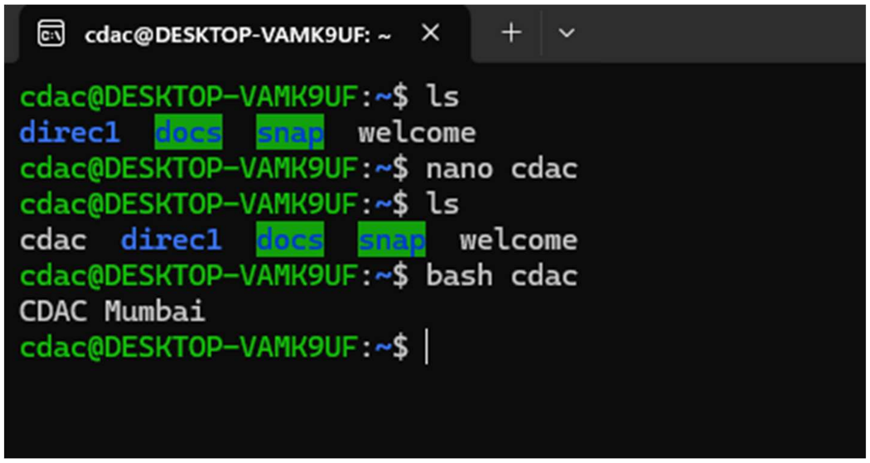


```
cdac@DESKTOP-VAMK9UF:~$ ls
direc1   docs   snap
cdac@DESKTOP-VAMK9UF:~$ nano welcome
cdac@DESKTOP-VAMK9UF:~$ ls
direc1   docs   snap   welcome
cdac@DESKTOP-VAMK9UF:~$ bash welcome
Hello World
cdac@DESKTOP-VAMK9UF:~$ 
```

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

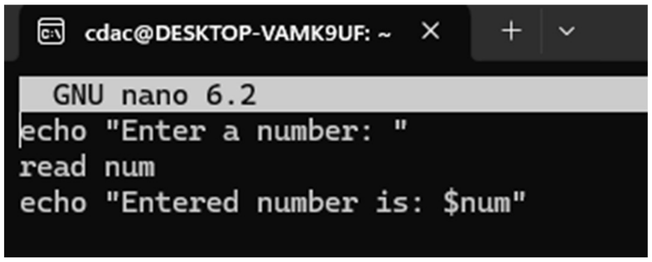


```
  GNU nano 6.2
name="CDAC Mumbai"
echo $name
```

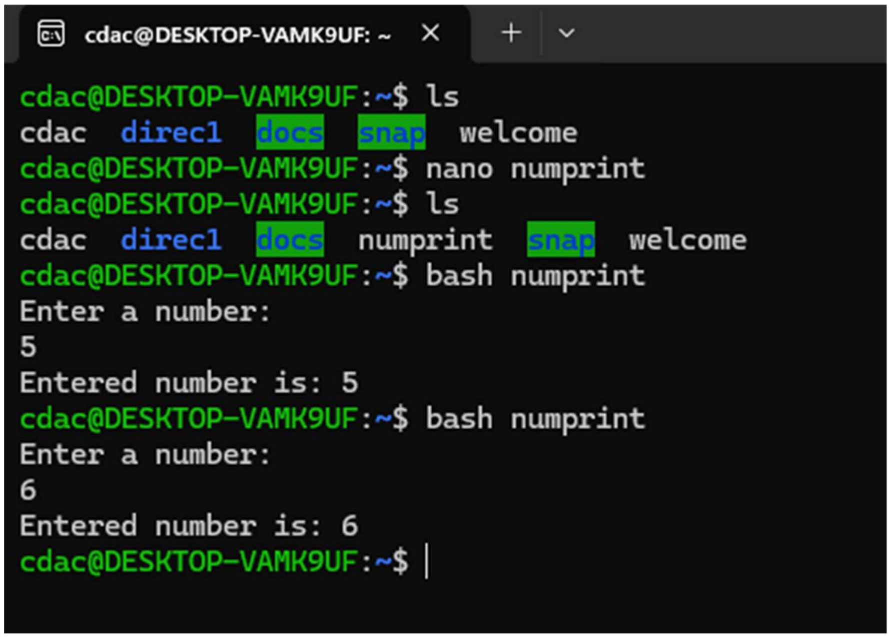


```
cdac@DESKTOP-VAMK9UF: ~
cdac@DESKTOP-VAMK9UF:~$ ls
direc1  docs  snap  welcome
cdac@DESKTOP-VAMK9UF:~$ nano cdac
cdac@DESKTOP-VAMK9UF:~$ ls
cdac  direc1  docs  snap  welcome
cdac@DESKTOP-VAMK9UF:~$ bash cdac
CDAC Mumbai
cdac@DESKTOP-VAMK9UF:~$ |
```

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

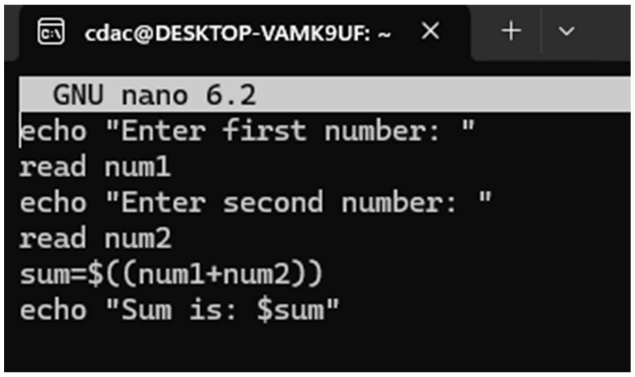


```
cdac@DESKTOP-VAMK9UF: ~
  GNU nano 6.2
echo "Enter a number: "
read num
echo "Entered number is: $num"
```

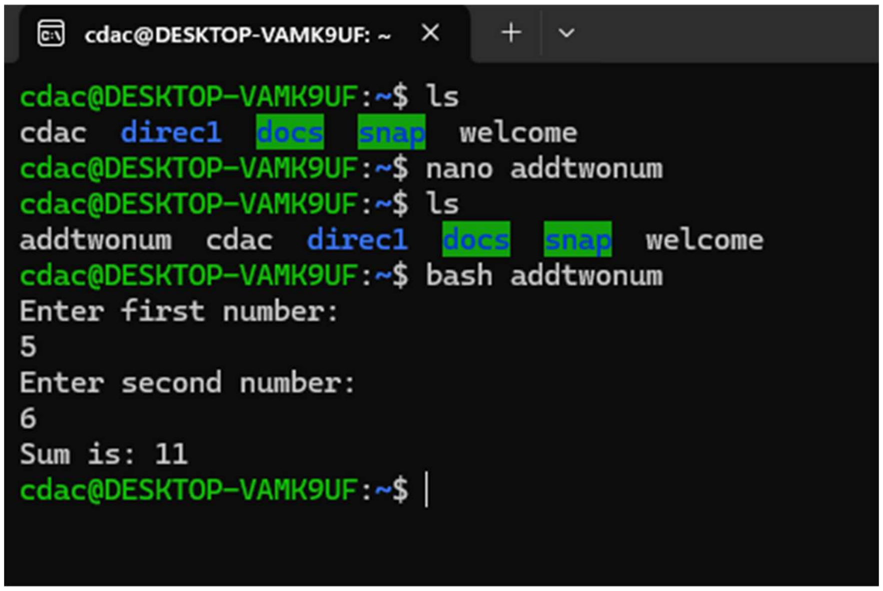


```
cdac@DESKTOP-VAMK9UF: ~
cdac@DESKTOP-VAMK9UF:~$ ls
cdac  direc1  docs  snap  welcome
cdac@DESKTOP-VAMK9UF:~$ nano numprint
cdac@DESKTOP-VAMK9UF:~$ ls
cdac  direc1  docs  numprint  snap  welcome
cdac@DESKTOP-VAMK9UF:~$ bash numprint
Enter a number:
5
Entered number is: 5
cdac@DESKTOP-VAMK9UF:~$ bash numprint
Enter a number:
6
Entered number is: 6
cdac@DESKTOP-VAMK9UF:~$ |
```

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

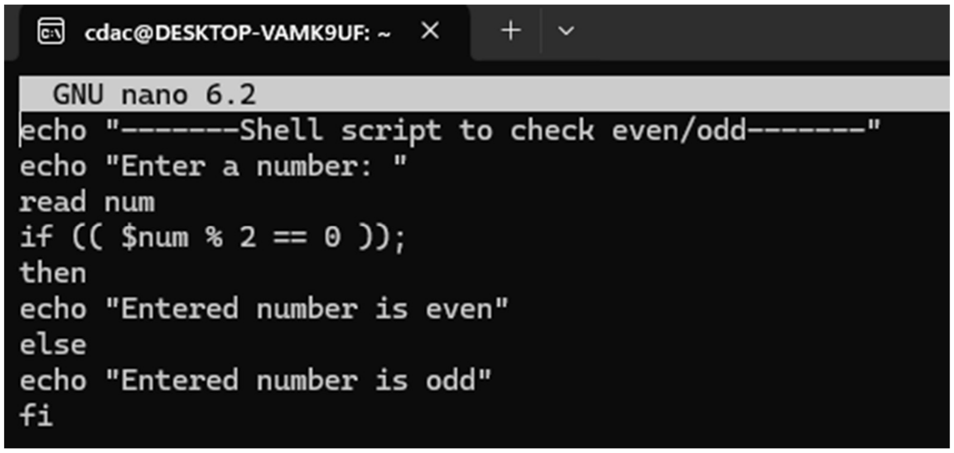


```
cdac@DESKTOP-VAMK9UF: ~
GNU nano 6.2
echo "Enter first number: "
read num1
echo "Enter second number: "
read num2
sum=$((num1+num2))
echo "Sum is: $sum"
```

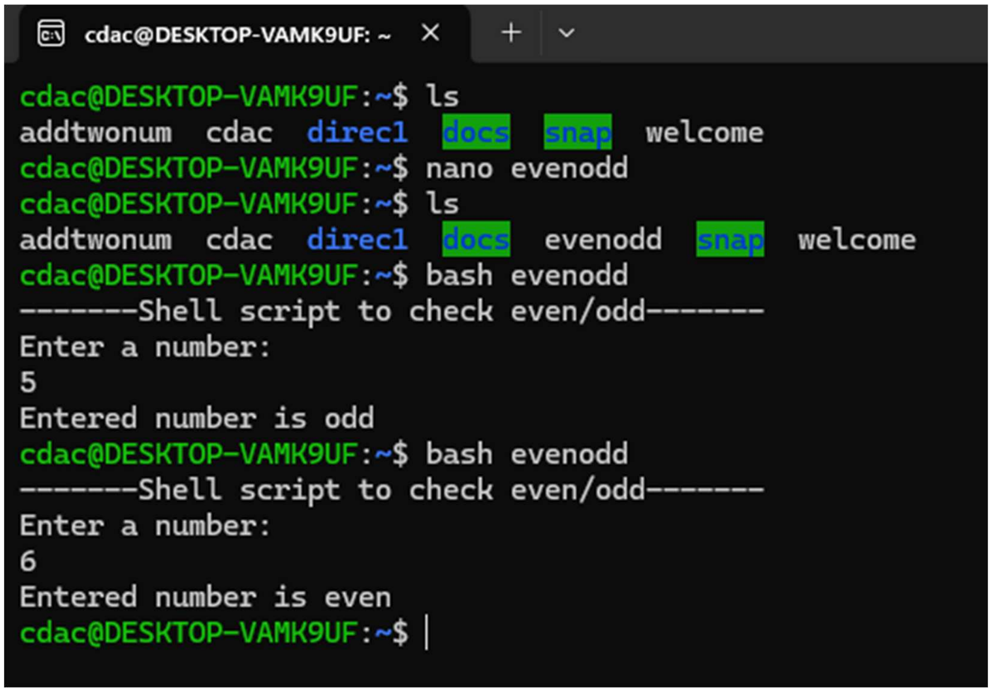


```
cdac@DESKTOP-VAMK9UF: ~
cdac@DESKTOP-VAMK9UF:~$ ls
cdac  direc1  docs  snap  welcome
cdac@DESKTOP-VAMK9UF:~$ nano addtwonum
cdac@DESKTOP-VAMK9UF:~$ ls
addtwonum  cdac  direc1  docs  snap  welcome
cdac@DESKTOP-VAMK9UF:~$ bash addtwonum
Enter first number:
5
Enter second number:
6
Sum is: 11
cdac@DESKTOP-VAMK9UF:~$
```

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

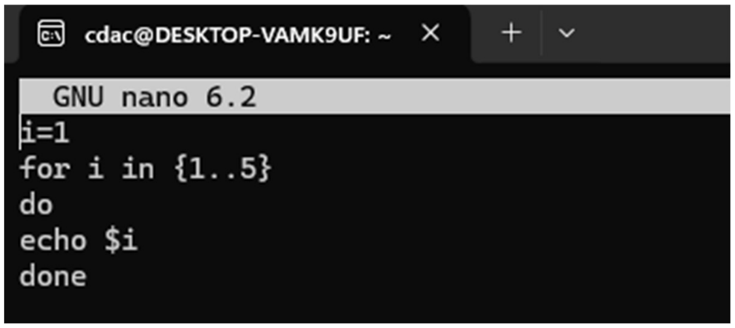


```
cdac@DESKTOP-VAMK9UF: ~
GNU nano 6.2
echo "-------Shell script to check even/odd-------"
echo "Enter a number: "
read num
if (( $num % 2 == 0 ));
then
echo "Entered number is even"
else
echo "Entered number is odd"
fi
```

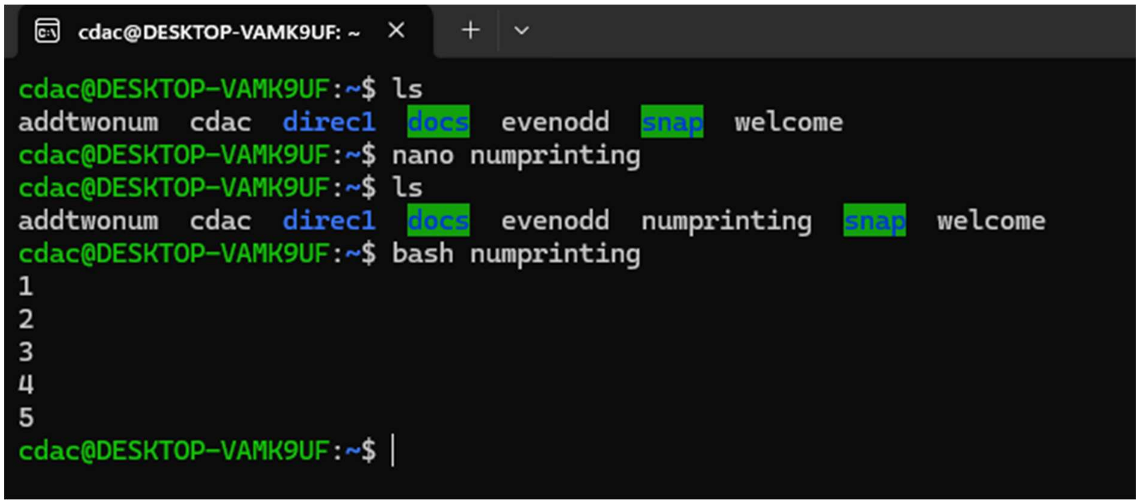


```
cdac@DESKTOP-VAMK9UF:~$ ls
addtwonum  cdac  direc1  docs  snap  welcome
cdac@DESKTOP-VAMK9UF:~$ nano evenodd
cdac@DESKTOP-VAMK9UF:~$ ls
addtwonum  cdac  direc1  docs  evenodd  snap  welcome
cdac@DESKTOP-VAMK9UF:~$ bash evenodd
-------Shell script to check even/odd-------
Enter a number:
5
Entered number is odd
cdac@DESKTOP-VAMK9UF:~$ bash evenodd
-------Shell script to check even/odd-------
Enter a number:
6
Entered number is even
cdac@DESKTOP-VAMK9UF:~$
```

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

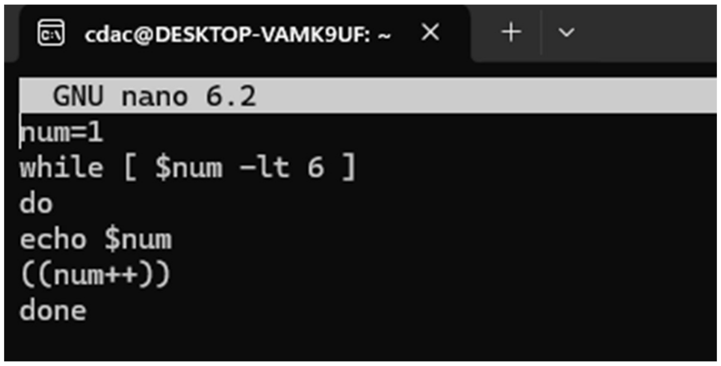


```
  GNU nano 6.2
i=1
for i in {1..5}
do
echo $i
done
```



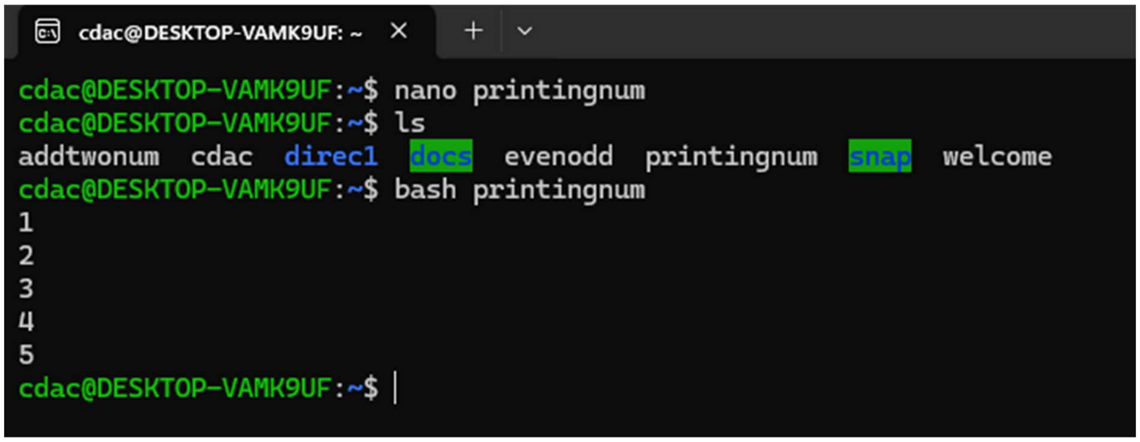
```
cdac@DESKTOP-VAMK9UF:~$ ls
addtwonum  cdac  direc1  docs  evenodd  snap  welcome
cdac@DESKTOP-VAMK9UF:~$ nano numprinting
cdac@DESKTOP-VAMK9UF:~$ ls
addtwonum  cdac  direc1  docs  evenodd  numprinting  snap  welcome
cdac@DESKTOP-VAMK9UF:~$ bash numprinting
1
2
3
4
5
cdac@DESKTOP-VAMK9UF:~$
```

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



cdac@DESKTOP-VAMK9UF: ~

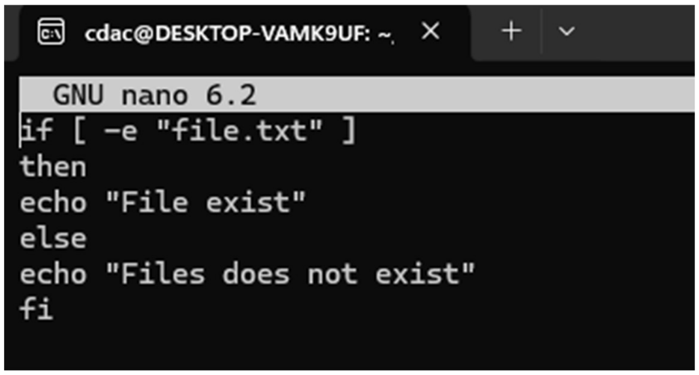
```
  GNU nano 6.2
num=1
while [ $num -lt 6 ]
do
echo $num
((num++))
done
```



cdac@DESKTOP-VAMK9UF: ~

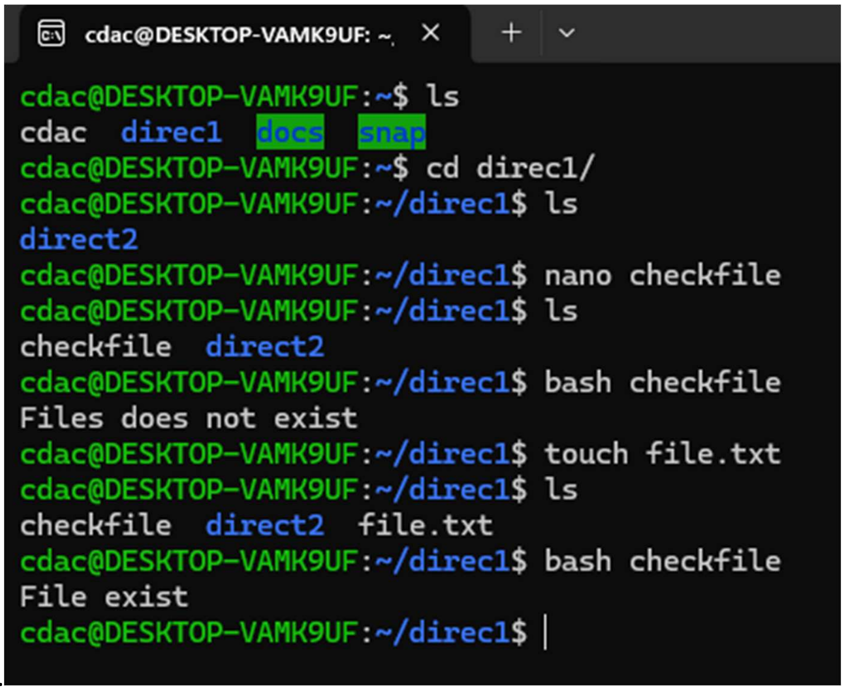
```
cdac@DESKTOP-VAMK9UF:~$ nano printingnum
cdac@DESKTOP-VAMK9UF:~$ ls
addtwonum  cdac  direc1  docs  evenodd  printingnum  snap  welcome
cdac@DESKTOP-VAMK9UF:~$ bash printingnum
1
2
3
4
5
cdac@DESKTOP-VAMK9UF:~$
```

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@DESKTOP-VAMK9UF: ~

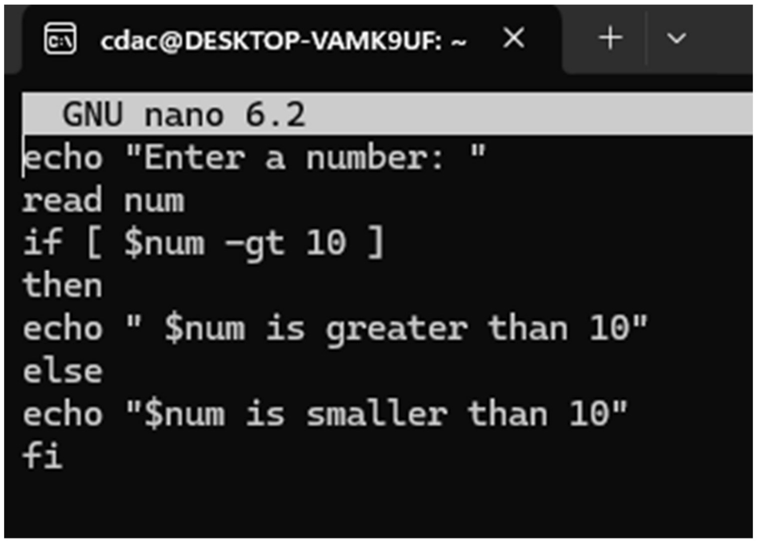
```
  GNU nano 6.2
if [ -e "file.txt" ]
then
echo "File exist"
else
echo "Files does not exist"
fi
```



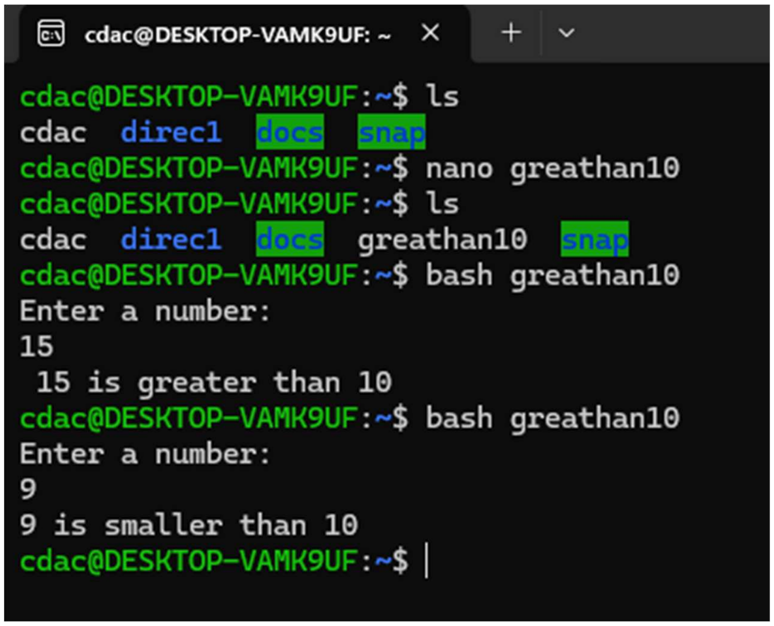
```
cdac@DESKTOP-VAMK9UF:~$ ls
cdac  direc1  docs  snap
cdac@DESKTOP-VAMK9UF:~$ cd direc1/
cdac@DESKTOP-VAMK9UF:~/direc1$ ls
direct2
cdac@DESKTOP-VAMK9UF:~/direc1$ nano checkfile
cdac@DESKTOP-VAMK9UF:~/direc1$ ls
checkfile  direct2
cdac@DESKTOP-VAMK9UF:~/direc1$ bash checkfile
Files does not exist
cdac@DESKTOP-VAMK9UF:~/direc1$ touch file.txt
cdac@DESKTOP-VAMK9UF:~/direc1$ ls
checkfile  direct2  file.txt
cdac@DESKTOP-VAMK9UF:~/direc1$ bash checkfile
File exist
cdac@DESKTOP-VAMK9UF:~/direc1$ 
```

.

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.

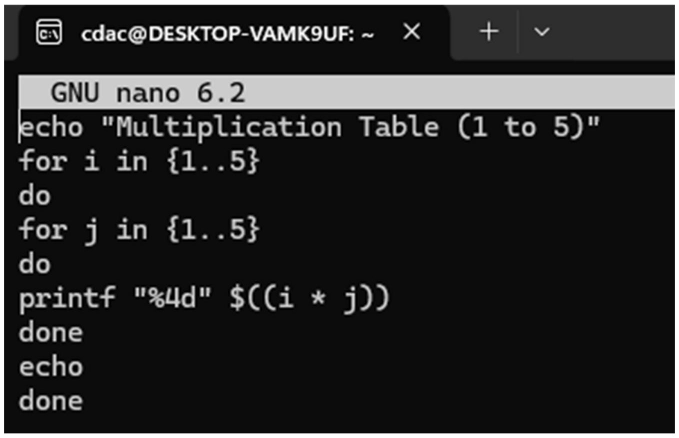


```
  GNU nano 6.2
echo "Enter a number: "
read num
if [ $num -gt 10 ]
then
echo " $num is greater than 10"
else
echo "$num is smaller than 10"
fi
```

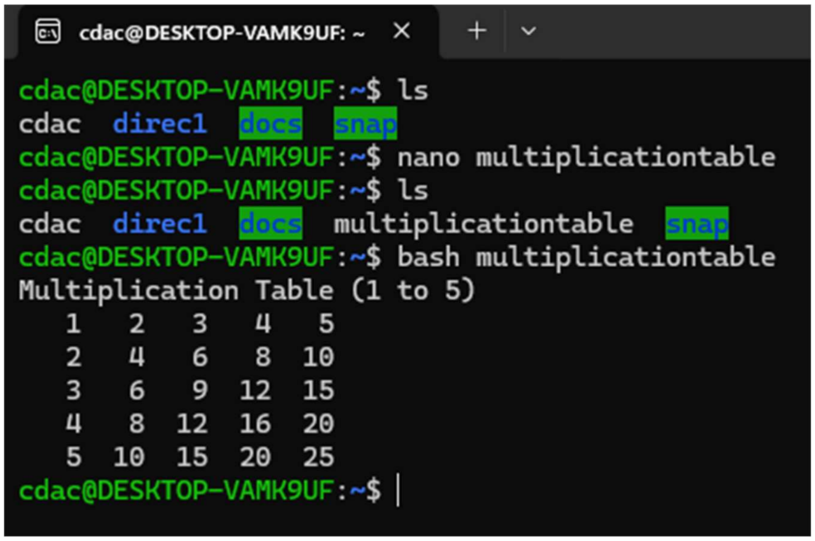


```
cdac@DESKTOP-VAMK9UF:~$ ls
cdac  direc1  docs  snap
cdac@DESKTOP-VAMK9UF:~$ nano greathan10
cdac@DESKTOP-VAMK9UF:~$ ls
cdac  direc1  docs  greathan10  snap
cdac@DESKTOP-VAMK9UF:~$ bash greathan10
Enter a number:
15
 15 is greater than 10
cdac@DESKTOP-VAMK9UF:~$ bash greathan10
Enter a number:
9
9 is smaller than 10
cdac@DESKTOP-VAMK9UF:~$
```

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.

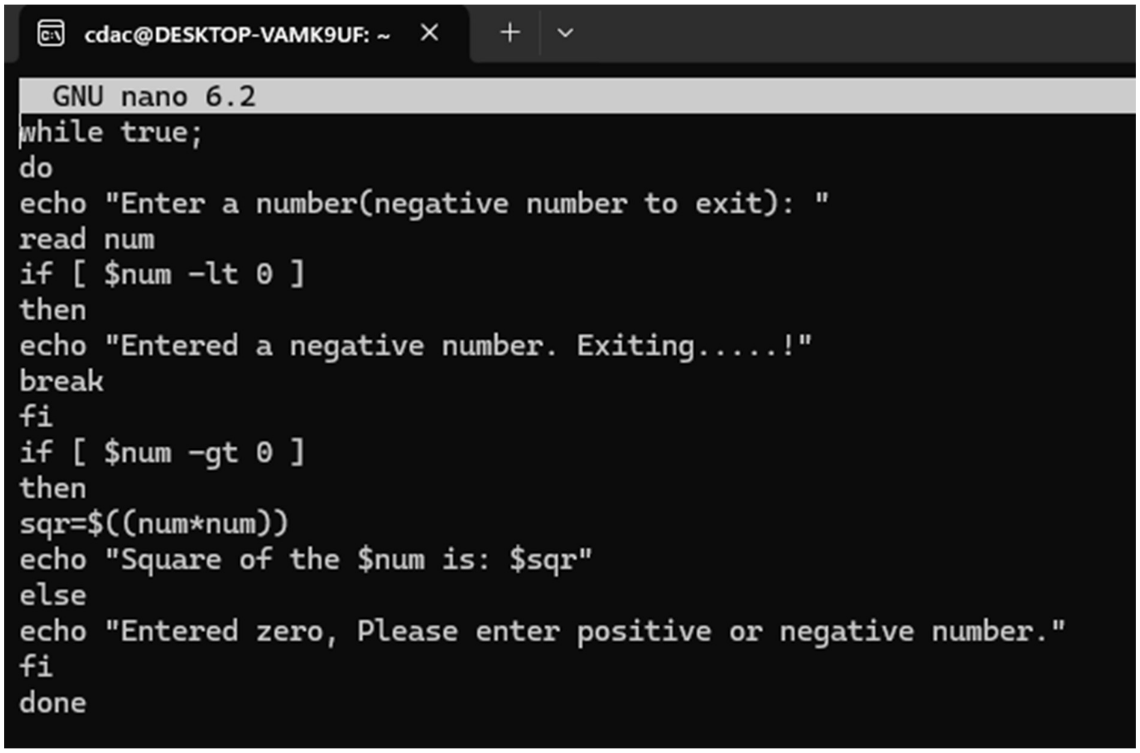


```
  GNU nano 6.2
echo "Multiplication Table (1 to 5)"
for i in {1..5}
do
for j in {1..5}
do
printf "%4d" $((i * j))
done
echo
done
```

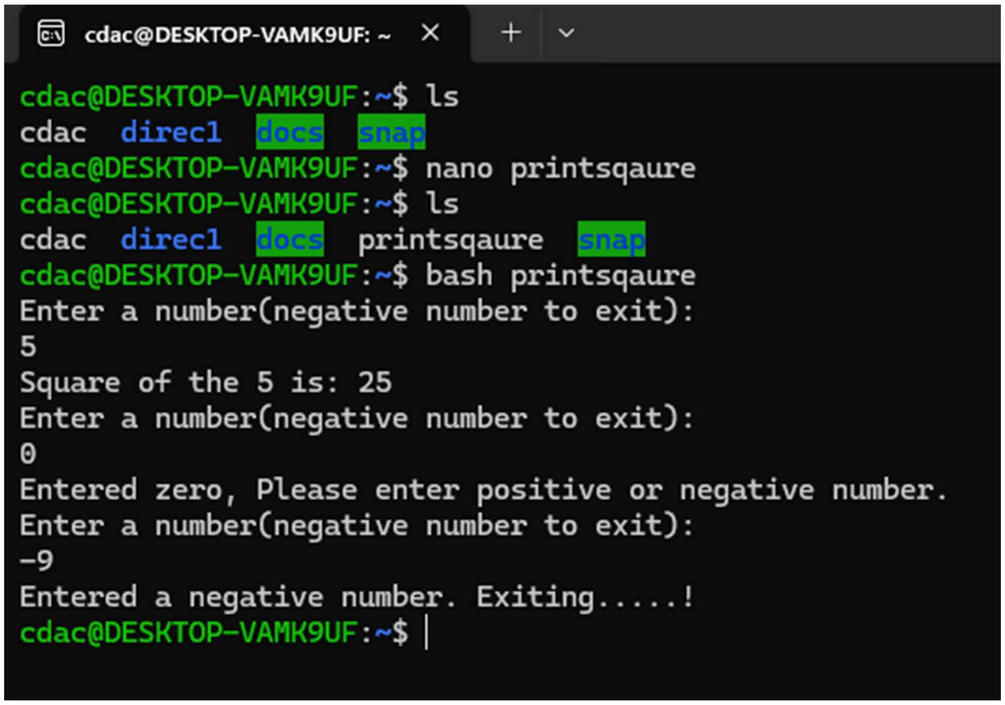


```
cdac@DESKTOP-VAMK9UF:~$ ls
cdac  direc1  docs  snap
cdac@DESKTOP-VAMK9UF:~$ nano multiplicationtable
cdac@DESKTOP-VAMK9UF:~$ ls
cdac  direc1  docs  multiplicationtable  snap
cdac@DESKTOP-VAMK9UF:~$ bash multiplicationtable
Multiplication Table (1 to 5)
   1   2   3   4   5
   2   4   6   8  10
   3   6   9  12  15
   4   8  12  16  20
   5  10  15  20  25
cdac@DESKTOP-VAMK9UF:~$
```

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
while true;
do
echo "Enter a number(negative number to exit): "
read num
if [ $num -lt 0 ]
then
echo "Entered a negative number. Exiting.....!"
break
fi
if [ $num -gt 0 ]
then
sqr=$((num*num))
echo "Square of the $num is: $sqr"
else
echo "Entered zero, Please enter positive or negative number."
fi
done
```



```
cdac@DESKTOP-VAMK9UF: ~

cdac@DESKTOP-VAMK9UF:~$ ls
cdac  direc1  docs  snap
cdac@DESKTOP-VAMK9UF:~$ nano printsqaure
cdac@DESKTOP-VAMK9UF:~$ ls
cdac  direc1  docs  printsqaure  snap
cdac@DESKTOP-VAMK9UF:~$ bash printsqaure
Enter a number(negative number to exit):
5
Square of the 5 is: 25
Enter a number(negative number to exit):
0
Entered zero, Please enter positive or negative number.
Enter a number(negative number to exit):
-9
Entered a negative number. Exiting.....!
cdac@DESKTOP-VAMK9UF:~$ |
```

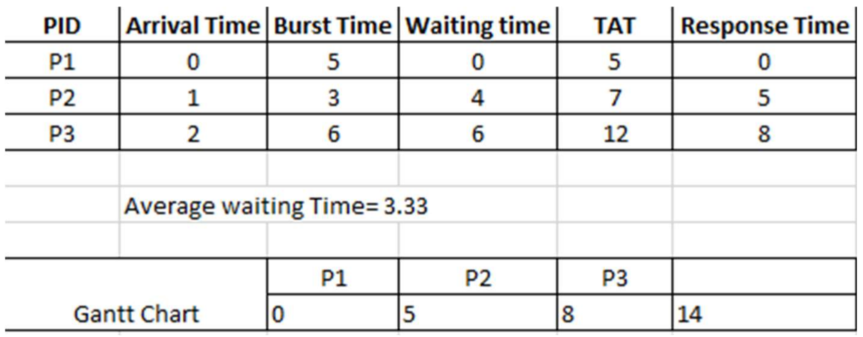
PART E :-

1. Consider the following processes with arrival times and burst times:

PID	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:-



PID	Arrival Time	Burst Time	Waiting time	TAT	Response Time
P1	0	5	0	5	0
P2	1	3	4	7	5
P3	2	6	6	12	8
	Average waiting Time= 3.33				
		P1	P2	P3	
Gantt Chart		0	5	8	14

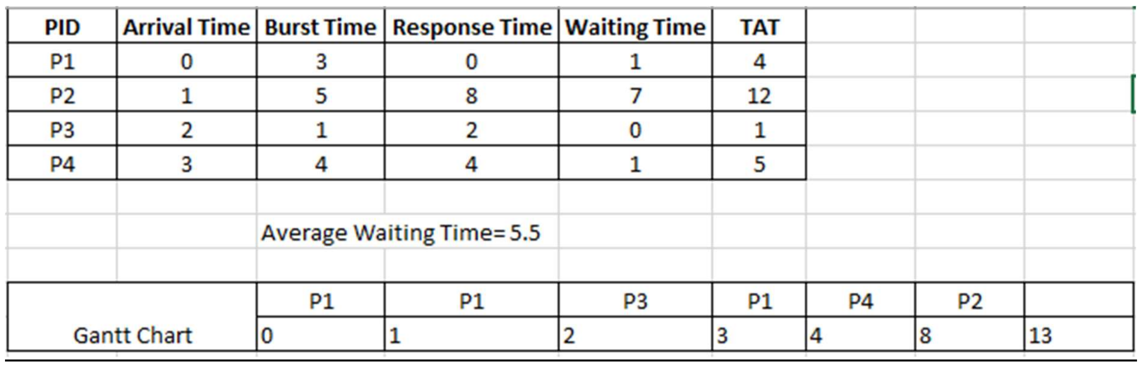
2. Consider the following processes with arrival times and burst times:

PID	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:-



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

Average Waiting Time= 5.5

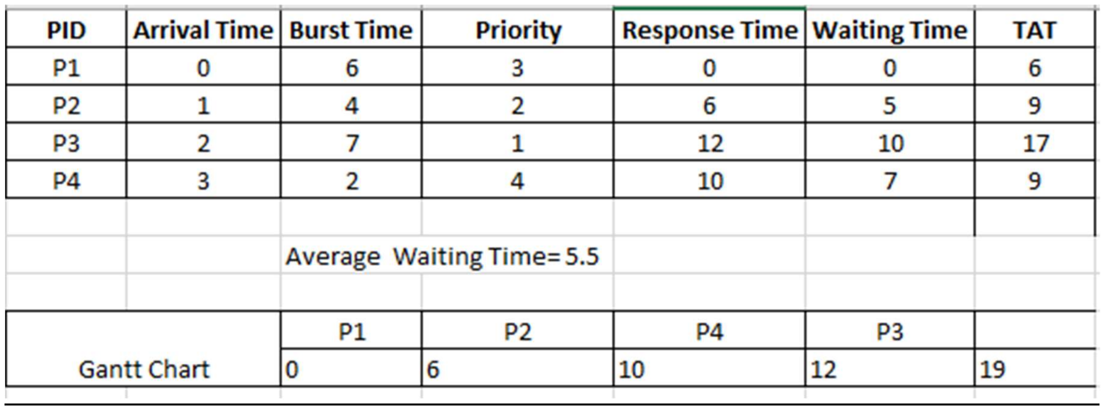
	P1	P1	P3	P1	P4	P2	
Gantt Chart	0	1	2	3	4	8	13

3. Consider the following processes with arrival times, burst times, and priorities (lower number indicates higher priority):

PID	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:-



PID	Arrival Time	Burst Time	Priority	Response Time	Waiting Time	TAT
P1	0	6	3	0	0	6
P2	1	4	2	6	5	9
P3	2	7	1	12	10	17
P4	3	2	4	10	7	9

Average Waiting Time= 5.5

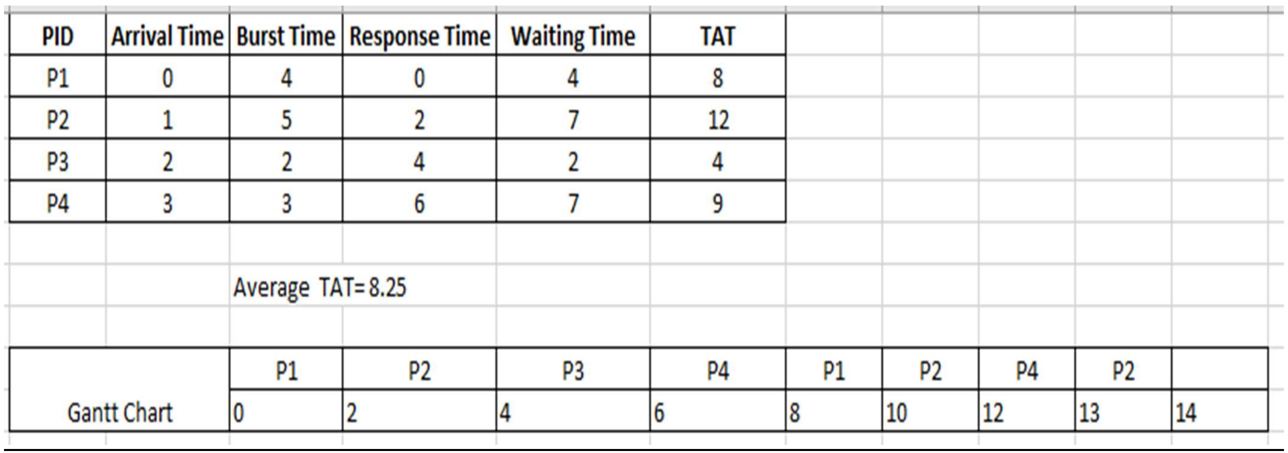
	P1	P2	P4	P3	
Gantt Chart	0	6	10	12	19

4. Consider the following processes with arrival times and burst times, and the time quantum for Round Robin scheduling is 2 units:

PID	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:-



PID	Arrival Time	Burst Time	Response Time	Waiting Time	TAT
P1	0	4	0	4	8
P2	1	5	2	7	12
P3	2	2	4	2	4
P4	3	3	6	7	9

Average TAT=8.25

Gantt Chart	P1	P2	P3	P4	P1	P2	P4	P2	
	0	2	4	6	8	10	12	13	14

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:-

```
#include <iostream>
#include <unistd.h>

int main() {
  int x = 5; // Initialize x with 5
  pid_t pid = fork(); // Create a child process

  if (pid < 0) {
    // Fork failed
    std::cout << "Fork failed!" << std::endl;
    return 1;
  } else if (pid == 0) {
    // Child process
    x = x + 1;
    std::cout << "Child Process: x = " << x << std::endl;
  } else {
    // Parent process
    x = x + 1;
    std::cout << "Parent Process: x = " << x << std::endl;
  }

  return 0;
}
```

<u>OUTPUT==</u>

```
Child Process: x = 6

Parent Process: x = 6
```

----------END----------