Concepts of Operating System
Assignment 2

Part A

What will the following commands do?

1. echo "Hello, World!"

=> Print Hello, World in terminal.

2. name="Productive"

=> Assign string “Productive” to variable name

3. touch file.txt

=> Create empty file.txt file

4. ls -a

=> List all files and directories

5. rm file.txt

=> Remove file.txt file6. cp file1.txt file2.txt

=> Copy file1.txt to file2.txt

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

=>Moves file.txt file to /path/to/directory/

8. chmod 755 script.sh

=> This command used to change permissions. Owner can read,write and execute and group and other only read and execute.

9. grep "pattern" file.txt

=> Searches for the specified pattern in file.txt

10. kill PID

=> Terminate the process with PID

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

=> Create directory and changes into it create empty file file.txt write hello word into file.txt and then display the content of file.txt

12. ls -l | grep ".txt"

=> This lists files in long format and filters the output to show only files with .txt in their names.

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

=> concatenates the contents of file1.txt and file2.txt, sorts the lines, and then removes any duplicate lines.

14. ls -l | grep "^d"

=> This lists files and directories in long format, then filters the output to show only directories

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

=> This searches recursively for the specified pattern in all files within the directory /path/to/directory/

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

=> concatenates the contents of file1.txt and file2.txt sorts the lines, and then shows only the duplicate lines.

16. chmod 644 file.txt

=> This sets the permissions read and write for the owner, read-only for others

17. cp -r source_directory destination_directory

=> This copies the entire source_directory and its contents to destination_directory

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

=> This searches for files with the .txt extension in the directory /path/to/search and all its subdirectories.

19. chmod u+x file.txt

=> This adds execute permissions for the owner of the file file.txt

20. echo $PATH

=>This prints the value of the PATH 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

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.

=>The correct command is chmod

2. cpy is used to copy files and directories.

=> The correct command is cp

3. mkfile is used to create a new file.

=> The correct command is touch

4. catx is used to concatenate files.

=>The correct command is cat

5. rn is used to rename files.

=>The correct command is mv

Part C

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

=> echo "Hello, World!"

Question2: Declare a variable named "name" and assign the value "CDAC Mumbai" to it. Print the value of the variable.

=> name="CDAC Mumbai"

echo $name

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

=> 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.

=> num1=5

num2=3

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".

=> echo "Enter a number:"

read number

if ((number % 2 == 0));

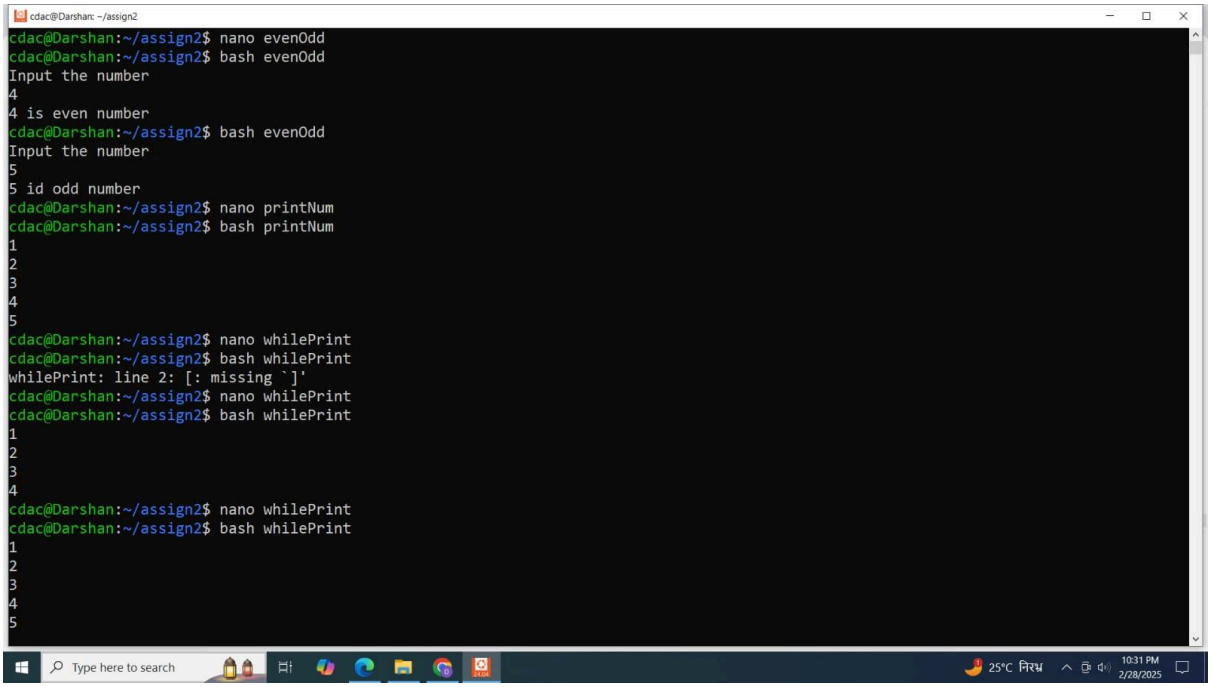
then

echo "$number is even number"

else

echo "$number is odd number"

fi



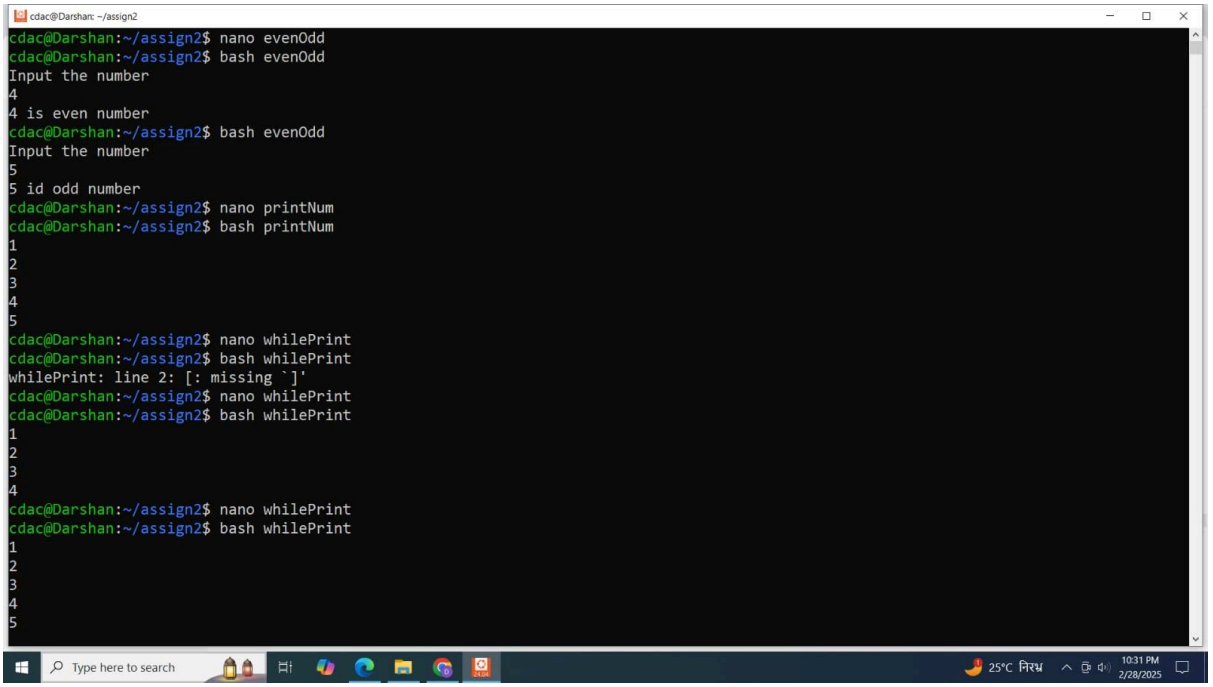
Question 6: Write a shell script that uses a for loop to print numbers from 1 to 5.

=> for i in {1..5}

do

echo $i

done



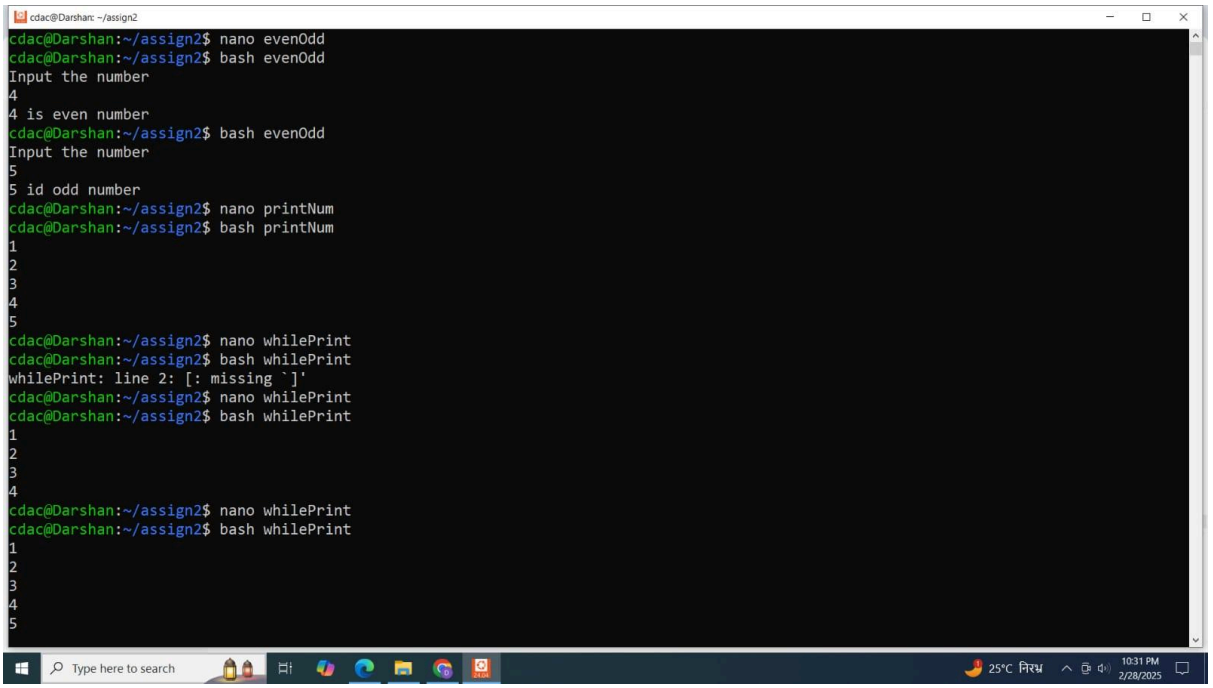
Question 7: Write a shell script that uses a while loop to print numbers from 1 to 5

=> 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".

=> if [-f "file.txt"];

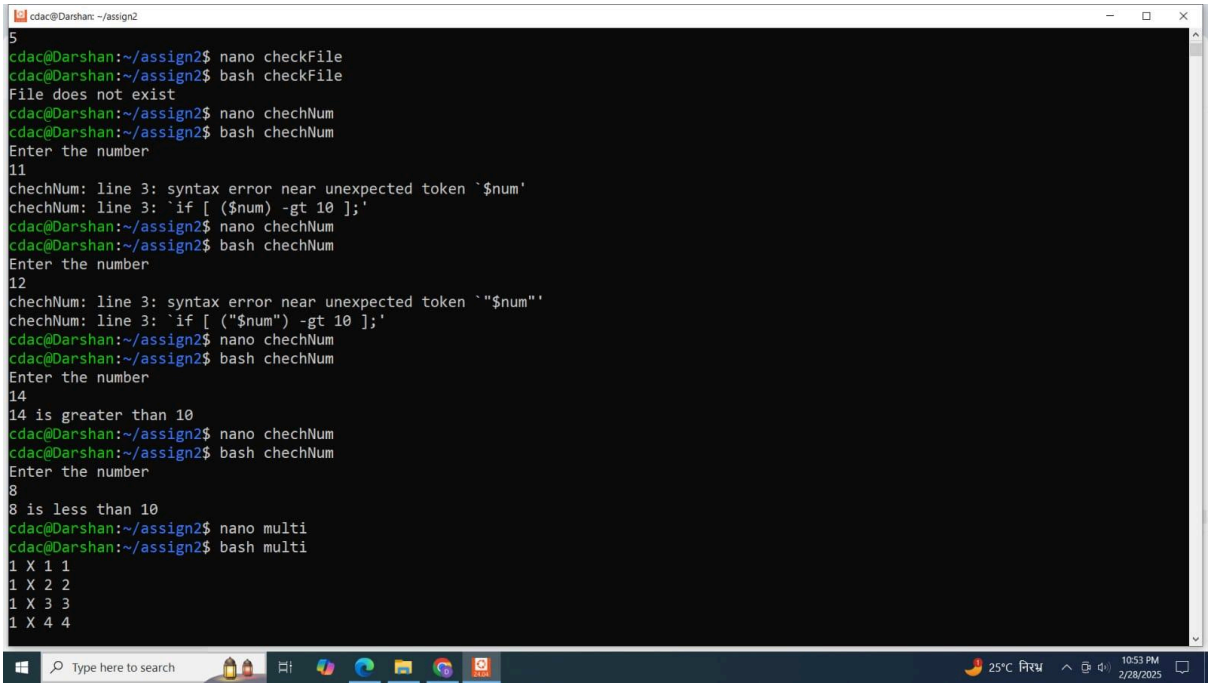
Then

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.

echo "Enter a number:"

read number

if [$number -gt 10];

then

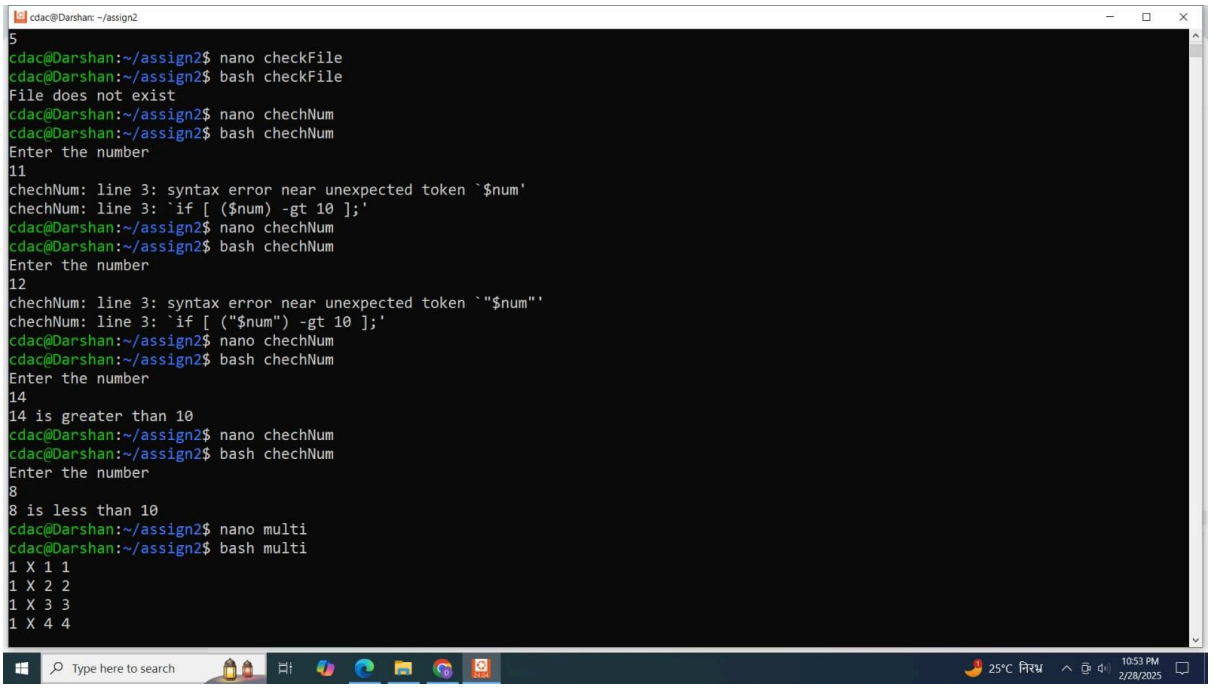
echo "$number is greater than 10."

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.

for i in {1..5}

do for j in {1..5}

echo "$i x $j = $((i*j))"

done

echo

done

=>

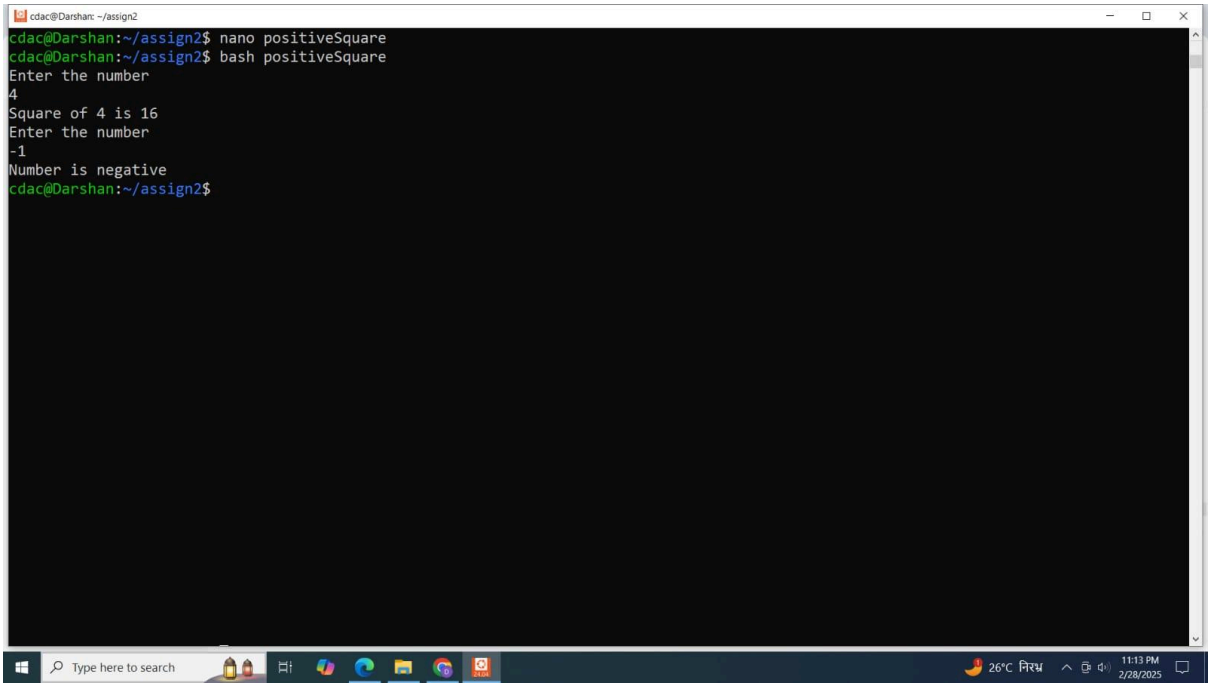
```
cdac@Darshan: ~/assign2

cdac@Darshan:~/assign2$ nano multi
cdac@Darshan:~/assign2$ bash multi
1 X 1 = 1
1 X 2 = 2
1 X 3 = 3
1 X 4 = 4
1 X 5 = 5
1 X 6 = 6
1 X 7 = 7
1 X 8 = 8
1 X 9 = 9
1 X 10 = 10

2 X 1 = 2
2 X 2 = 4
2 X 3 = 6
2 X 4 = 8
2 X 5 = 10
2 X 6 = 12
2 X 7 = 14
2 X 8 = 16
2 X 9 = 18
2 X 10 = 20

3 X 1 = 3
3 X 2 = 6
3 X 3 = 9
3 X 4 = 12
3 X 5 = 15
3 X 6 = 18
3 X 7 = 21
```

```
cdac@Darshan: ~/assign2

4 X 1 = 4
4 X 2 = 8
4 X 3 = 12
4 X 4 = 16
4 X 5 = 20
4 X 6 = 24
4 X 7 = 28
4 X 8 = 32
4 X 9 = 36
4 X 10 = 40

5 X 1 = 5
5 X 2 = 10
5 X 3 = 15
5 X 4 = 20
5 X 5 = 25
5 X 6 = 30
5 X 7 = 35
5 X 8 = 40
5 X 9 = 45
5 X 10 = 50

cdac@Darshan:~/assign2$
```

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:"
 read number
 if [ $number -lt 0 ];
then
  echo"Number is negative"
  break
 fi
 square=$((number * number))
 echo "The square of $number is $square"
done
```



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.

Process	AT	BT	WT	TAT		
p1	0	5	0	5		
p2	1	3	4	7		
p3	2	6	6	12		
			AWT= 3.34			
		Gantt chart	p1	p2	p3	
			0	5	8	14
		Ans.	Avg waiting time= 3.34			

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.

Process	AT	BT		TAT				
p1	0	3		4				
p2	1	5		8				
p3	2	1		0				
p4	3	4		10				
		Gantt chart	p1	p3	p1	p2	p4	
			0	1	2	3	8	13
		Ans.	ATAT = 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.

Process	AT	BT	Priority	WT	TAT					
p1	0	6	3	6	13					
p2	1	4	1	13	18					
p3	2	7	4	1	7					
p4	3	2	2	10	12					
		Gantt chart	p1	p1	p3	p3	p1	p4	p2	
			0	1	2	3	9	13	15	19
		Ans.	AWT=7.5							

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.

Process	AT	BT		TAT							
p1	0	4		10							
p2	1	5		13							
p3	2	2		4							
p4	3	3		10							
		Gantt chart	p1	p2	p3	p4	p1	p2	p4	p2	
			0	2	4	6	8	10	12	13	14
		Ans.	ATAT= 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?

=> Final Values:

Parent processes= 6

Child processes= 6