CDAC MUMBAI

Concepts of Operating System
Assignment 2

Part A

What will the following commands do?

- **echo "Hello, World!"**
  Prints "Hello, World!" to terminal

- **name="Productive"**
  Assigns the value "Productive" to the variable name

- **touch file.txt**
  Creates a new, empty file named file.txt in the current directory

- **ls -a**
  List all files including hidden files

- **rm file.txt**
  Deletes the file named file.txt from the current directory.

- **cp file1.txt file2.txt**
  Copies the contents of file1.txt to file2.txt

- **mv file.txt /path/to/directory/**
  Moves the file file.txt to the specified directory

- **chmod 755 script.sh**
  7 Gives read , write ,execute permission to user
  5 gives read , execute permission to group
  5 gives read , execute permission to other

- **grep "pattern" file.txt**
  Search the word pattern into file.txt

- **kill PID**
  Terminate the process by using process id

- **mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt**
  mkdir mydir: Creates a directory named mydir.
  cd mydir: Changes the current directory to mydir.
  touch file.txt: Creates a new, empty file named file.txt.
  echo "Hello, World!" > file.txt: Writes "Hello, World!" to file.txt.
  cat file.txt: Displays the contents of file.txt, which will be "Hello, World!".

- **ls -l | grep ".txt"**
  It shows all files with .txt extension in current directory

- **cat file1.txt file2.txt | sort | uniq**
  Cat file1.txt file2.txt concatenate file and sort them by alphabetically
  if there is duplicate value present then it shows only one unique value

- **ls -l | grep "^d"**
  It shows all subdirectory in current directory

- **grep -r "pattern" /path/to/directory/**
  search pattern text recursively through all directories and subdirectories
  under the specified path.

- **cat file1.txt file2.txt | sort | uniq –d**
  cat file1.txt file2.txt : Concatenates the contents of file1.txt and file2.txt.
  sort : Sorts the combined contents.
  uniq -d : Displays only the duplicate lines.

- **chmod 644 file.txt**
  6 gives permission to owner of the file can read and write the file.
  4 gives permission to associated with the file can only read the file.
  4 gives permission to Others (everyone else) can only read the file.

- **cp -r source_directory destination_directory**
  Copy all the files and directory from source directory to destination directory

- **find /path/to/search -name "*.txt"**
  Find files with .txt extension

- **chmod u+x file.txt**
  Give permission to owner to execute file.txt

- **echo $PATH**
  Displays the current user's PATH environment variable, which lists directories where the system looks for executable files.

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 (cd is used to change the current directory)

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
   Correct : Chmod u+x file.txt

2. **cpy** is used to copy files and directories. Incorrect
   Correct : cp

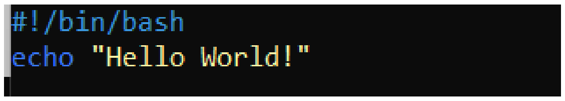
3. **mkfile** is used to create a new file. Incorrect
   Correct : Touch file.txt or nano file.txt

4. **catx** is used to concatenate files. Incorrect
   Correct : cat file1.txt

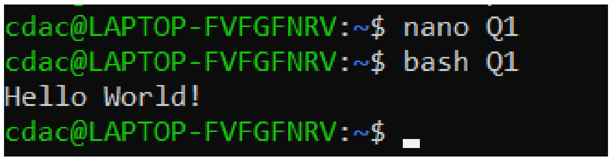
5. **rn** is used to rename files. Incorrect
   Correct : mv old_name.txt new_name.txt

Part C

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

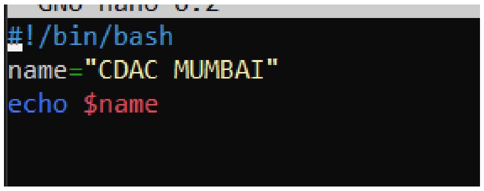


```
#!/bin/bash
echo "Hello World!"
```

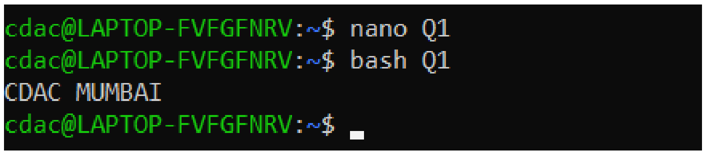


```
cdac@LAPTOP-FVFGFNRV:~$ nano Q1
cdac@LAPTOP-FVFGFNRV:~$ bash Q1
Hello World!
cdac@LAPTOP-FVFGFNRV:~$
```

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

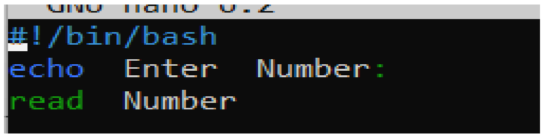


```
#!/bin/bash
name="CDAC MUMBAI"
echo $name
```

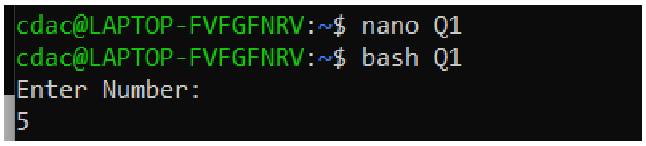


```
cdac@LAPTOP-FVFGFNRV:~$ nano Q1
cdac@LAPTOP-FVFGFNRV:~$ bash Q1
CDAC MUMBAI
cdac@LAPTOP-FVFGFNRV:~$
```

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

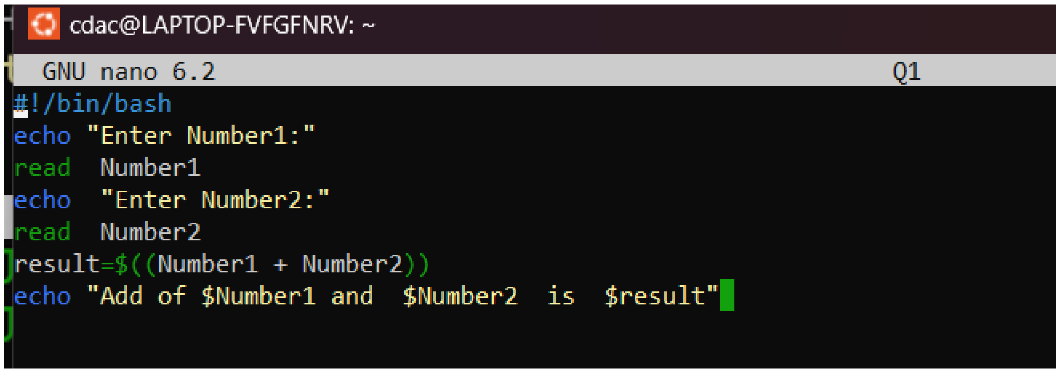


```
#!/bin/bash
echo  Enter  Number:
read  Number
```

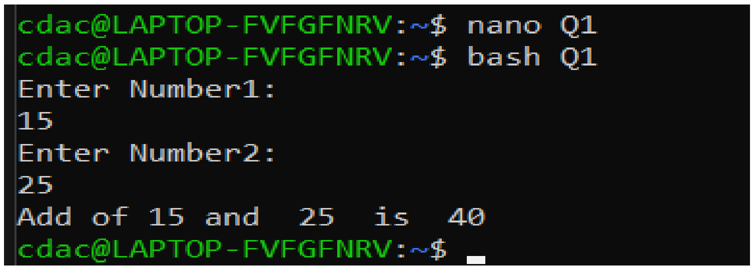


```
cdac@LAPTOP-FVFGFNRV:~$ nano Q1
cdac@LAPTOP-FVFGFNRV:~$ bash Q1
Enter Number:
5
```

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

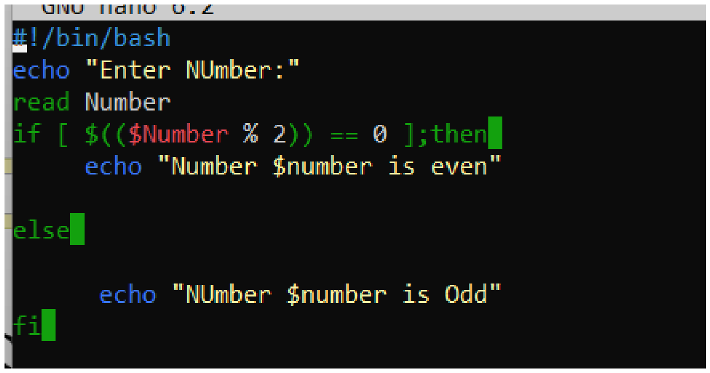


```
cdac@LAPTOP-FVFGFNRV: ~
 GNU nano 6.2                                                    Q1
#!/bin/bash
echo "Enter Number1:"
read  Number1
echo  "Enter Number2:"
read  Number2
result=$((Number1 + Number2))
echo "Add of $Number1 and  $Number2  is  $result"
```

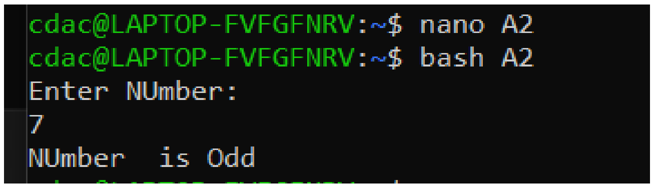


```
cdac@LAPTOP-FVFGFNRV:~$ nano Q1
cdac@LAPTOP-FVFGFNRV:~$ bash Q1
Enter Number1:
15
Enter Number2:
25
Add of 15 and  25  is  40
cdac@LAPTOP-FVFGFNRV:~$
```

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

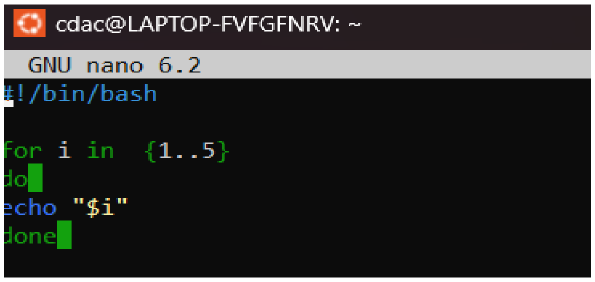


```
#!/bin/bash
echo "Enter NUmber:"
read Number
if [ $(($Number % 2)) == 0 ];then
    echo "Number $number is even"

else

      echo "NUmber $number is Odd"
fi
```

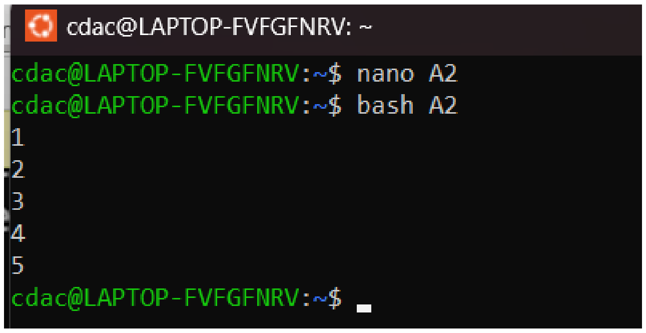


```
cdac@LAPTOP-FVFGFNRV:~$ nano A2
cdac@LAPTOP-FVFGFNRV:~$ bash A2
Enter NUmber:
7
NUmber  is Odd
```

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

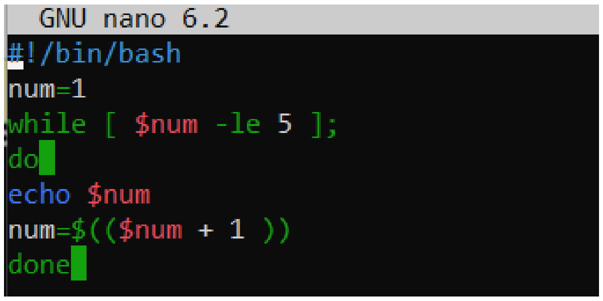


```
cdac@LAPTOP-FVFGFNRV: ~
 GNU nano 6.2
#!/bin/bash

for i in  {1..5}
do
echo "$i"
done
```

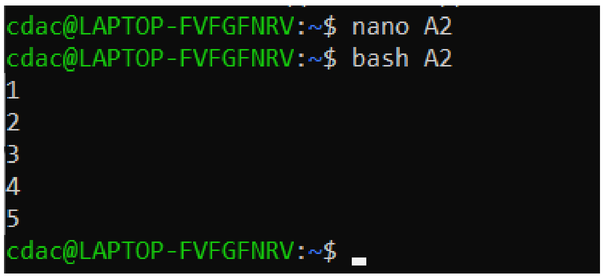


```
cdac@LAPTOP-FVFGFNRV: ~
cdac@LAPTOP-FVFGFNRV:~$ nano A2
cdac@LAPTOP-FVFGFNRV:~$ bash A2
1
2
3
4
5
cdac@LAPTOP-FVFGFNRV:~$
```

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

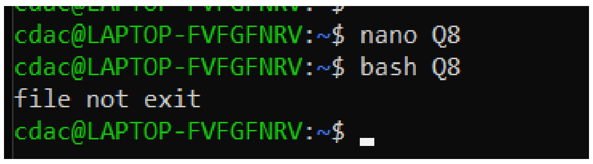


```
 GNU nano 6.2
#!/bin/bash
num=1
while [ $num -le 5 ];
do
echo $num
num=$(($num + 1 ))
done
```

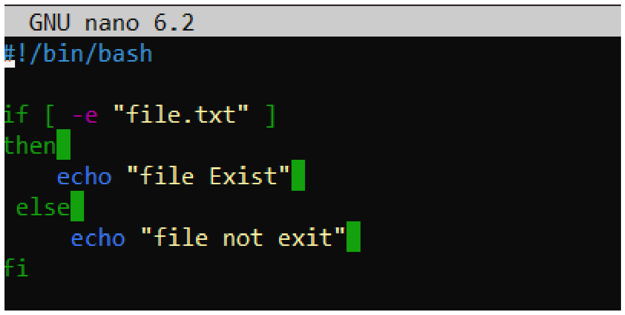


```
cdac@LAPTOP-FVFGFNRV:~$ nano A2
cdac@LAPTOP-FVFGFNRV:~$ bash A2
1
2
3
4
5
cdac@LAPTOP-FVFGFNRV:~$
```

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".

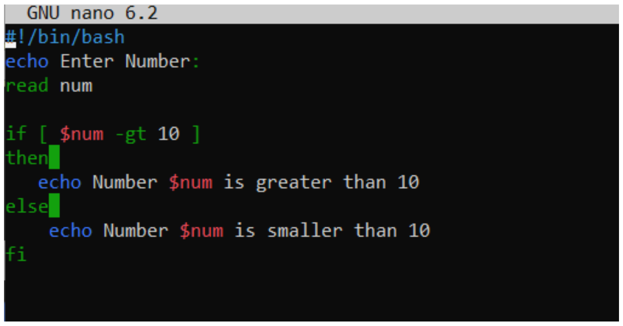


```
cdac@LAPTOP-FVFGFNRV:~$ nano Q8
cdac@LAPTOP-FVFGFNRV:~$ bash Q8
file not exit
cdac@LAPTOP-FVFGFNRV:~$
```

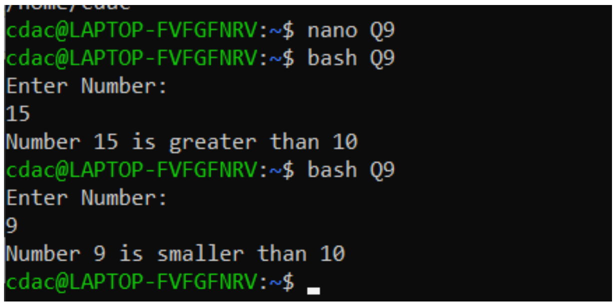


```
 GNU nano 6.2
#!/bin/bash

if [ -e "file.txt" ]
then
    echo "file Exist"
 else
     echo "file not exit"
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.

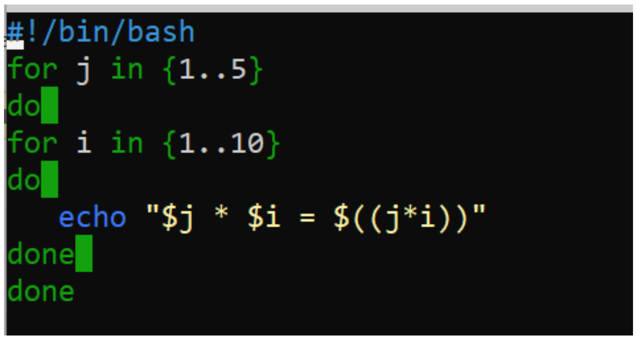


```
 GNU nano 6.2
#!/bin/bash
echo Enter Number:
read num

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



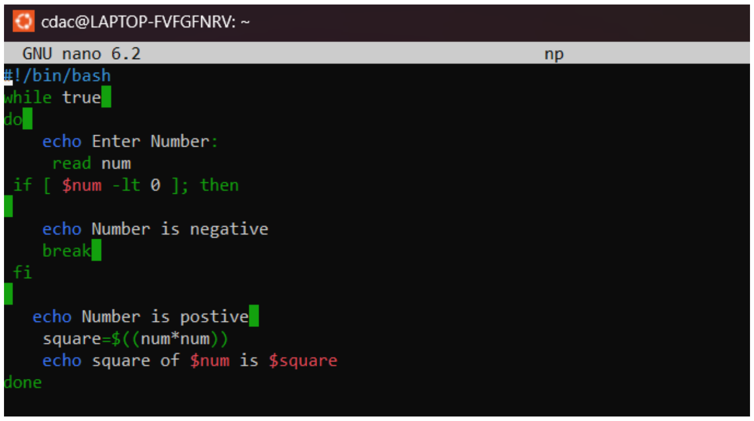
```
cdac@LAPTOP-FVFGFNRV:~$ nano Q9
cdac@LAPTOP-FVFGFNRV:~$ bash Q9
Enter Number:
15
Number 15 is greater than 10
cdac@LAPTOP-FVFGFNRV:~$ bash Q9
Enter Number:
9
Number 9 is smaller than 10
cdac@LAPTOP-FVFGFNRV:~$
```

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.

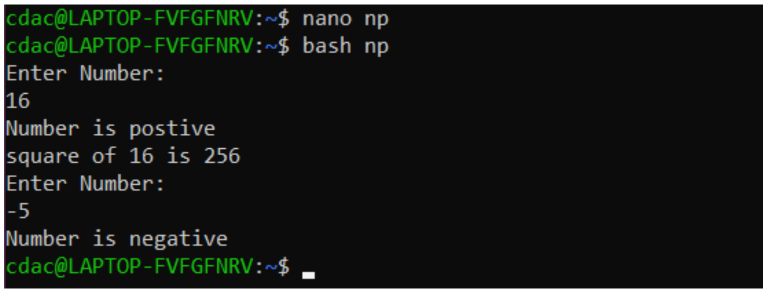


```
#!/bin/bash
for j in {1..5}
do
for i in {1..10}
do
    echo "$j * $i = $((j*i))"
done
done
```

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.



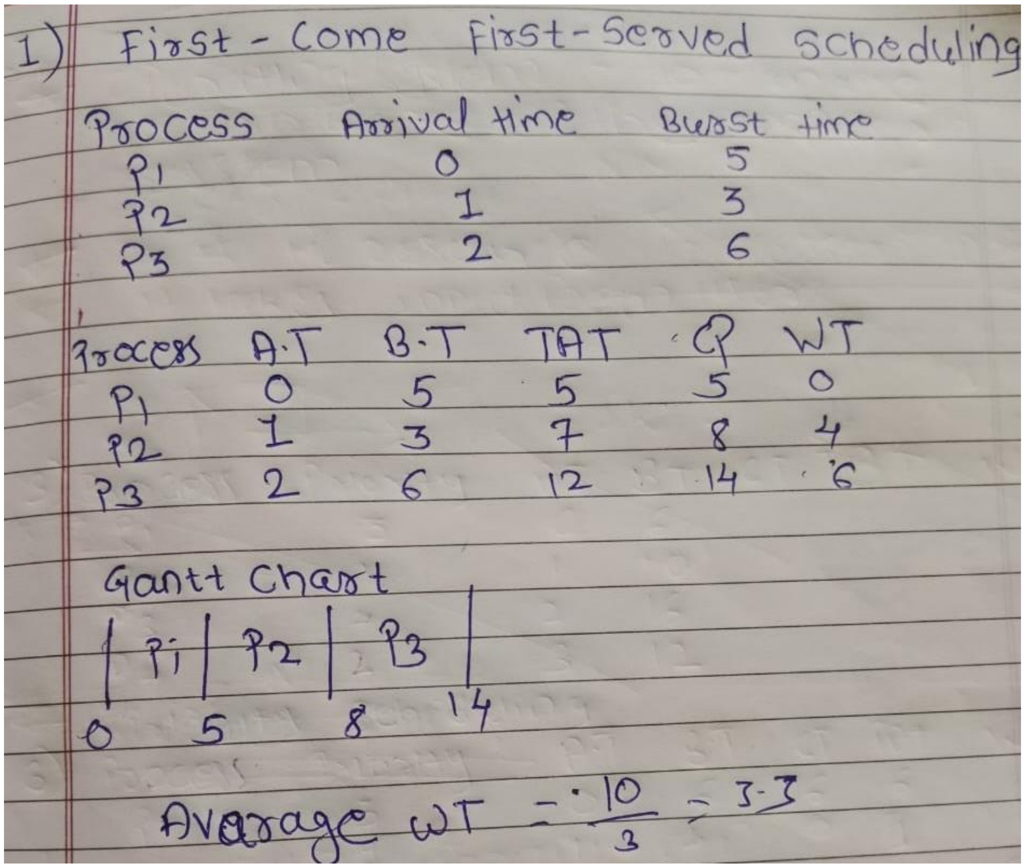
```
cdac@LAPTOP-FVFGFNRV: ~
  GNU nano 6.2                                            np
#!/bin/bash
while true
do
    echo Enter Number:
     read num
 if [ $num -lt 0 ]; then

    echo Number is negative
    break
 fi

   echo Number is postive
    square=$((num*num))
    echo square of $num is $square
done
```



```
cdac@LAPTOP-FVFGFNRV:~$ nano np
cdac@LAPTOP-FVFGFNRV:~$ bash np
Enter Number:
16
Number is postive
square of 16 is 256
Enter Number:
-5
Number is negative
cdac@LAPTOP-FVFGFNRV:~$ _
```

Part E

1. Consider the following processes with arrival times and burst times:
2. Consider the following processes with arrival times and burst times:
3. Consider the following processes with arrival times, burst times, and priorities (lower number indicates higher priority):
4. Consider the following processes with arrival times and burst times, and the time quantum for Round Robin scheduling is 2 units:
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?



1) First - Come First-Served Scheduling

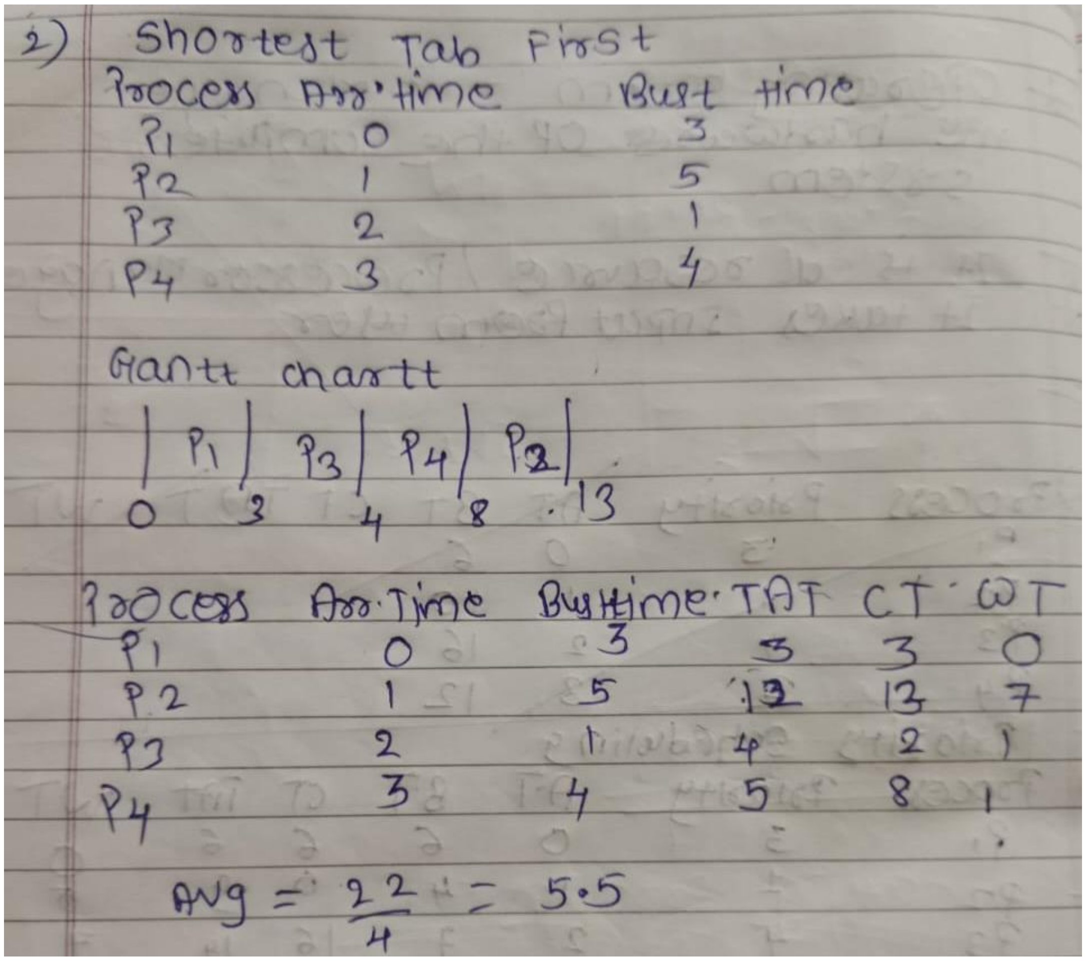
Process	Arrival time	Burst time
P1	0	5
P2	1	3
P3	2	6

Process	A.T	B.T	TAT	CP	WT
P1	0	5	5	5	0
P2	1	3	7	8	4
P3	2	6	12	14	6

Gantt Chart

```
| P1 | P2 | P3 |
0    5    8    14
```

Avarage WT = $\frac{10}{3}$ = 3.3



2) Shortest Jab First

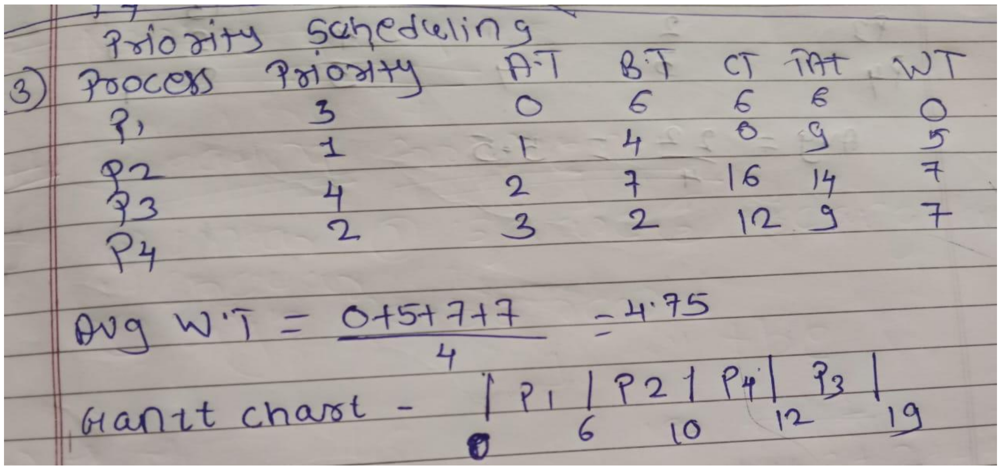
Process	Arr' time	Bust time
P1	0	3
P2	1	5
P3	2	1
P4	3	4

Gantt chartt

```
| P1 | P3 | P4 | P2 |
0    3    4    8    13
```

Process	Arr. Time	Bus time	TAT	CT	WT
P1	0	3	3	3	0
P.2	1	5	12	13	7
P3	2	1	4	2	1
P4	3	4	5	8	1

Avg = $\frac{22}{4}$ = 5.5



Priority Scheduling

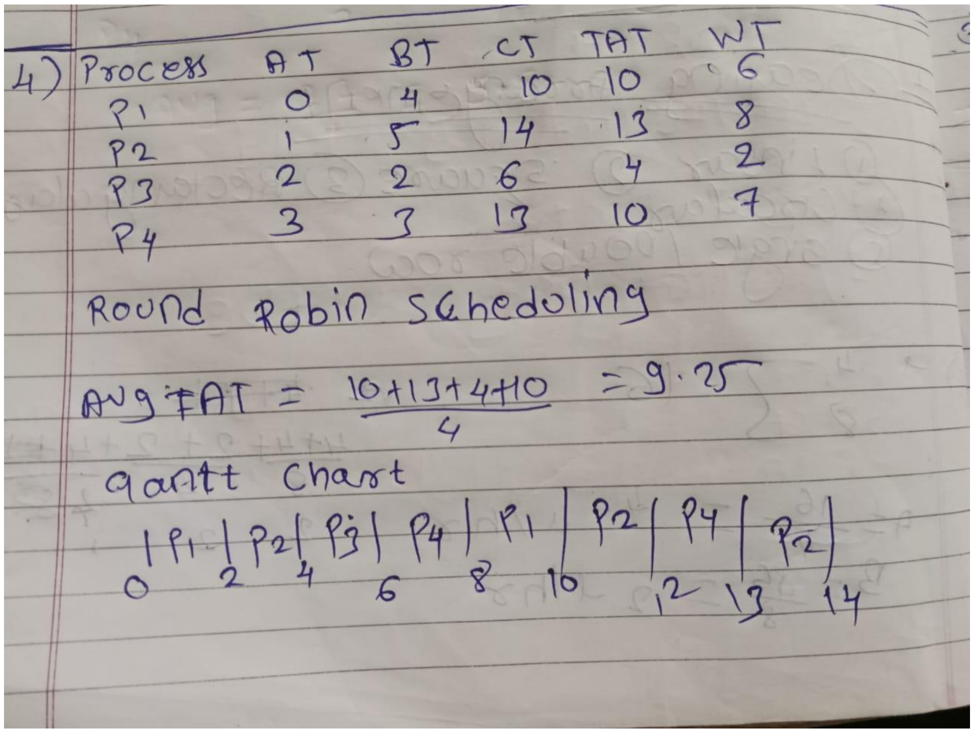
3)

Process	Priority	AT	BT	CT	TAT	WT
P1	3	0	6	6	6	0
P2	1	1	4	8	9	5
P3	4	2	7	16	14	7
P4	2	3	2	12	9	7

$$\text{Avg W.T} = \frac{0+5+7+7}{4} = 4.75$$

Gantt chart - | P1 | P2 | P4 | P3 |

0 6 10 12 19



4)

Process	AT	BT	CT	TAT	WT
P1	0	4	10	10	6
P2	1	5	14	13	8
P3	2	2	6	4	2
P4	3	3	13	10	7

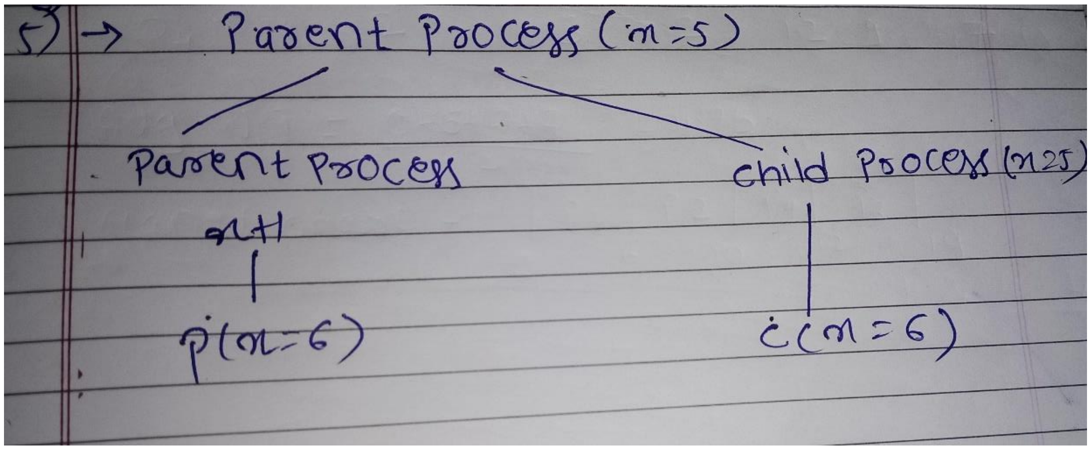
Round Robin Scheduling

$$\text{Avg TAT} = \frac{10+13+4+10}{4} = 9.25$$

gantt Chart

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

0 2 4 6 8 10 12 13 14


5) →
Parent Process (n=5)
Parent Process
child Process (n=5)
n+1
p(n=6)
c(n=6)