# CDAC MUMBAI

## Concepts of Operating System

### Assignment 2

## Part A

### What will the following commands do?

- echo "Hello, World!"

```
cdac@LAPTOP-NQM5IGSU:~$ echo "Hello,World!"
Hello,World!
```

The **echo** command in Linux is used to display a string to the terminal.

- name=”Productive”

```
cdac@LAPTOP-NQM5IGSU:~$ name="Productive"
cdac@LAPTOP-NQM5IGSU:~$ echo $name
Productive
```

It creates variable named “name” and stores string “Productive” in it.

- touch file.txt

```
cdac@LAPTOP-NQM5IGSU:~$ touch file.txt
cdac@LAPTOP-NQM5IGSU:~$ ls
A   AAA  ABC   LinuxAssignment  aa   aaaa   cdac1     dir3  dir5  duplicate.txt  file11.txt  file2.zip  fruit.txt  numbers.txt  practice  s2.sh
AA  AB   ABCD  a               aaa  aaaaa  data.txt  dir4  dir6  file.txt       file2.txt   file3.txt  g1.txt     output.txt   s1.sh     s3.sh
```

Create empty text file with the name file.txt

- ls –a

```
cdac@LAPTOP-NQM5IGSU:~$ ls -a
.              .bashrc     .motd_shown              A    ABC             aa     cdac1     dir5          file11.txt  fruit.txt   practice
..             .cache      .profile                 AA   ABCD            aaa    data.txt  dir6          file2.txt   g1.txt      s1.sh
.bash_history  .landscape  .sudo_as_admin_successful  AAA  LinuxAssignment  aaaa   dir3      duplicate.txt  file2.zip   numbers.txt  s2.sh
.bash_logout   .lesshst    .viminfo                 AB   a               aaaaa  dir4      file.txt       file3.txt   output.txt  s3.sh
```

List all the files and directories in given directory including hidden ones (. , ..).

- rm file.txt

```
cdac@LAPTOP-NQM5IGSU:~$ rm file.txt
cdac@LAPTOP-NQM5IGSU:~$ ls
A   AAA  ABC   LinuxAssignment  aa   aaaa   cdac1     dir3  dir5  duplicate.txt  file2.txt  file3.txt  g1.txt      output.txt  s1.sh  s3.sh
AA  AB   ABCD  a               aaa  aaaaa  data.txt  dir4  dir6  file11.txt     file2.zip  fruit.txt  numbers.txt  practice   s2.sh
```

To remove file

- cp file1.txt file2.txt

```
cdac@LAPTOP-NQM5IGSU:~$ cat > file1.txt
Hello World!
cdac@LAPTOP-NQM5IGSU:~$ cat file2.txt
hello linux world
this is linux, please use proper commands
ctrl + d => save and next
cdac@LAPTOP-NQM5IGSU:~$ cp file1.txt file2.txt
cdac@LAPTOP-NQM5IGSU:~$ cat file2.txt
Hello World!
```

copies the contents of file1.txt into a new file named file2.txt

If the file2.txt already exits then it overwritten without warning (unless you use the –i option, it gives warning before overwritten)

- mv file1.txt /path/to/directory/

```
cdac@LAPTOP-NQM5IGSU:~$ cat > file1.txt
This is the file1.txt
cdac@LAPTOP-NQM5IGSU:~$ mv file1.txt ./practice/
cdac@LAPTOP-NQM5IGSU:~$ cd practice
cdac@LAPTOP-NQM5IGSU:~/practice$ ls
docs1  docs1.zip  file1.txt  file2.txt  my_docs
```

  Move the file from the current directory to destination directory.

- chmod 755 script.sh

```
cdac@LAPTOP-NQM5IGSU:~/practice$ touch script.sh
cdac@LAPTOP-NQM5IGSU:~/practice$ ls -l script.sh
-rw-r--r-- 1 cdac cdac 0 Aug 20 22:38 script.sh
cdac@LAPTOP-NQM5IGSU:~/practice$ chmod 755 script.sh
cdac@LAPTOP-NQM5IGSU:~/practice$ ls -l script.sh
-rwxr-xr-x 1 cdac cdac 0 Aug 20 22:38 script.sh
```

  Changes the permissions of script.sh so that :
  - Owner : read , write and execute(rwx)
  - Group : read and execute(r-x)
  - Others : read and execute(r-x)

- grep “pattern” file.txt

```
cdac@LAPTOP-NQM5IGSU:~/practice$ cat > file.txt
pattern
Pattern
pattern
PaTTern
pattern
PATTERN
cdac@LAPTOP-NQM5IGSU:~/practice$ grep "pattern" file.txt
pattern
pattern
pattern
```

  Searches the specific word in the given file

- kill PID

```
cdac@LAPTOP-NQM5IGSU:~/LinuxAssignment$ sleep 1000 &
[1] 455
cdac@LAPTOP-NQM5IGSU:~/LinuxAssignment$ kill 455
cdac@LAPTOP-NQM5IGSU:~/LinuxAssignment$ ps
    PID TTY          TIME CMD
    303 pts/0    00:00:00 bash
    456 pts/0    00:00:00 ps
[1]+  Terminated              sleep 1000
```

  This command used to kill the process.

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

```
cdac@LAPTOP-NQM5IGSU:~$ mkdir mydir && cd mydir && touch file.txt && echo "Hello,World!" > file.txt && cat file.txt
Hello,World!
cdac@LAPTOP-NQM5IGSU:~/mydir$ ls
file.txt
```

Create the directory named mydir, goes inside it, create an empty file **file.txt**, write "Hello, World" into it, and then displays the file content.

- ls –l | ".txt"

```
cdac@LAPTOP-NQM5IGSU:~$ ls -l | grep ".txt"
-rw-r--r-- 1 cdac cdac  308 Aug 20 20:54 data.txt
-rw-r--r-- 1 cdac cdac  109 Aug 20 21:52 duplicate.txt
-rw-r--r-- 1 cdac cdac  117 Aug 19 15:18 file11.txt
-rw-r--r-- 1 cdac cdac   23 Aug 20 22:30 file2.txt
-rw-r--r-- 1 cdac cdac   16 Aug 19 14:50 file3.txt
-rw-r--r-- 1 cdac cdac   85 Aug 20 21:56 fruit.txt
-rw-r--r-- 1 cdac cdac   45 Aug 20 06:51 g1.txt
-rw-r--r-- 1 cdac cdac   51 Aug 20 06:44 numbers.txt
-rw-r--r-- 1 cdac cdac   13 Aug 20 21:49 output.txt
```

List the files in the long format in the current directory and filters out the output to show only the files that ends with '.txt'

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

```
cdac@LAPTOP-NQM5IGSU:~$ cat file1.txt file2.txt | sort | uniq
Hello, from the CDAC Kharghar!
This is the text file.
```

Displays the content of file1.txt and file2.txt, sort the lines and remove duplicates.

- ls -l | grep "^d"

```
cdac@LAPTOP-NQM5IGSU:~$ ls -l | grep "^d"
drwxr-xr-x 4 cdac cdac 4096 Aug 20 20:42 LinuxAssignment
drwxr-xr-x 2 cdac cdac 4096 Aug 19 14:40 cdac1
drwxr-xr-x 2 cdac cdac 4096 Aug 19 14:43 dir3
drwxr-xr-x 2 cdac cdac 4096 Aug 19 14:43 dir4
drwxr-xr-x 2 cdac cdac 4096 Aug 19 14:43 dir5
drwxr-xr-x 2 cdac cdac 4096 Aug 19 14:43 dir6
drwxr-xr-x 2 cdac cdac 4096 Aug 21 06:30 mydir
drwxr-xr-x 4 cdac cdac 4096 Aug 20 22:42 practice
```

List only the directories in the current directory by filtering **ls –l** for the output start with d(which represents directory)

"^d" search only the line starts with d not  d present anywhere else.

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

```
cdac@LAPTOP-NQM5IGSU:~$ grep -r "pattern" ./practice/
./practice/file.txt:pattern
./practice/file.txt:pattern
./practice/file.txt:pattern
```

searches recursively for the "pattern" for the given directory.

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

```
cdac@LAPTOP-NQM5IGSU:~$ cat file1.txt file2.txt | sort | uniq -d
Hello, from the CDAC Kharghar!
This is the text file.
```

shows only the duplicate lines in both file1.txt and file2.txt

- chmod 644 file.txt

```
cdac@LAPTOP-NQM5IGSU:~$ touch file.txt
cdac@LAPTOP-NQM5IGSU:~$ ls -l file.txt
-rw-r--r-- 1 cdac cdac 0 Aug 21 07:17 file.txt
cdac@LAPTOP-NQM5IGSU:~$ chmod 644 file.txt
cdac@LAPTOP-NQM5IGSU:~$ ls -l file.txt
-rw-r--r-- 1 cdac cdac 0 Aug 21 07:17 file.txt
```

Changes the permissions of file.txt so that :

- Owner : read and write (rw-)
- Group : only read (r--)
- Others : only read (r--)

- cp -r source_directory destination_directory

```
cdac@LAPTOP-NQM5IGSU:~$ cp mydir ./mydir1/ -r
cdac@LAPTOP-NQM5IGSU:~$ cd mydir1
cdac@LAPTOP-NQM5IGSU:~/mydir1$ ls
mydir
```

copy entire source directory into the destination directory.

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

```
cdac@LAPTOP-NQM5IGSU:~$ find ./practice/ -name "*.txt"
./practice/file2.txt
./practice/file.txt
./practice/file1.txt
./practice/my_docs/docs1/duplicates.txt
./practice/my_docs/docs1/numbers.txt
./practice/my_docs/docs1/file1.txt
./practice/my_docs/docs1/input.txt
./practice/my_docs/docs1/data.txt
./practice/docs1/file1.txt
```

Searches all files ending with “*.txt” in the given path and its sub directories.

- chmod u+x file.txt

```
cdac@LAPTOP-NQM5IGSU:~/mydir1/mydir$ chmod u+x file.txt
cdac@LAPTOP-NQM5IGSU:~/mydir1/mydir$ ls -l file.txt
-rwxr--r-- 1 cdac cdac 13 Aug 21 07:31 file.txt
```

Add the execute permission to the owner.

- echo $PATH

```
cdac@LAPTOP-NQM5IGSU:~/mydir1/mydir$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/usr/lib/wsl/lib:/mnt/c/Program Files/Common Files/Oracle/Java/javapath:/mnt/c/Windows/system32:/mnt/c/Windows:/mnt/c/Windows/System32/Wbem:/mnt/c/Windows/System32/WindowsPowerShell/v1.0/:/mnt/c/Windows/System32/OpenSSH/:/mnt/c/Program Files (x86)/NVIDIA Corporation/PhysX/Common:/mnt/c/Program Files/NVIDIA Corporation/NVIDIA NvDLISR:/mnt/c/WINDOWS/system32:/mnt/c/WINDOWS:/mnt/c/WINDOWS/System32/Wbem:/mnt/c/WINDOWS/System32/WindowsPowerShell/v1.0/:/mnt/c/WINDOWS/System32/OpenSSH/:/mnt/c/Program Files/nodejs/:/mnt/c/Program Files/Amazon/AWSCLIV2/:/mnt/c/Program Files/Docker/Docker/resources/bin:/mnt/c/Program Files/Git/cmd:/mnt/c/TDM-GCC-64/bin:/mnt/c/MinGW/bin:/mnt/c/Users/LENOVO/AppData/Local/Programs/Python/Python312/Scripts/:/mnt/c/Users/LENOVO/AppData/Local/Programs/Python/Python312/:/mnt/c/Users/LENOVO/AppData/Local/Programs/Python/Launcher/:/mnt/c/Program Files/MySQL/MySQL Shell 8.0/bin/:/mnt/c/Users/LENOVO/AppData/Local/Microsoft/WindowsApps:/mnt/c/Users/LENOVO/AppData/Roaming/npm:/mnt/c/Users/LENOVO/AppData/Local/Programs/Microsoft VS Code/bin:/mnt/c/Program Files/Git/bin:/mnt/c/Program Files (x86)/Nmap:/mnt/d/All Semester Data/Sem 6/SEPM/apache-maven-3.9.6/bin:/mnt/d/anacondanew/condabin:/mnt/d/anacondanew/Scripts:/mnt/d/anacondanew/Library/bin:/mnt/c/Users/LENOVO/AppData/Local/Programs/cursor/resources/app/bin:/snap/bin
```

  prints the value of the environment variable 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**
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. 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)
2. cpy is used to copy files and directories.
   **Incorrect** (correct : cp)
3. mkfile is used to create a new file.
   **Incorrect** (correct : touch)

4. catx is used to concatenate files.
   **Incorrect** (correct : cat)
5. rn is used to rename files.
   **Incorrect** (correct : mv oldname newname)

# Part C

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

```
cdac@LAPTOP-NQM5IGSU:~$ vi helloworld.sh
cdac@LAPTOP-NQM5IGSU:~$ cat helloworld.sh
#!/bin/bash
echo "Hello World!"

cdac@LAPTOP-NQM5IGSU:~$ ls -l helloworld.sh
-rw-r--r-- 1 cdac cdac 33 Aug 21 08:24 helloworld.sh
cdac@LAPTOP-NQM5IGSU:~$ chmod +x helloworld.sh
cdac@LAPTOP-NQM5IGSU:~$ ls -l helloworld.sh
-rwxr-xr-x 1 cdac cdac 33 Aug 21 08:24 helloworld.sh
cdac@LAPTOP-NQM5IGSU:~$ ./helloworld.sh
Hello World!
```

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

```
cdac@LAPTOP-NQM5IGSU:~$ vi printname.sh
cdac@LAPTOP-NQM5IGSU:~$ cat printname.sh
#!/bin/bash
name="CDAC Mumbai"
echo "$name"

cdac@LAPTOP-NQM5IGSU:~$ chmod +x printname.sh
cdac@LAPTOP-NQM5IGSU:~$ ./printname.sh
CDAC Mumbai
```

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

```
cdac@LAPTOP-NQM5IGSU:~$ vi printnumber.sh
cdac@LAPTOP-NQM5IGSU:~$ cat printnumber.sh
#!/bin/bash
echo "Enter a number"
read number
echo "$number"

cdac@LAPTOP-NQM5IGSU:~$ chmod +x printnumber.sh
cdac@LAPTOP-NQM5IGSU:~$ ./printnumber.sh
Enter a number
25
25
```

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

```
cdac@LAPTOP-NQM5IGSU:~$ vi addition.sh
cdac@LAPTOP-NQM5IGSU:~$ cat addition.sh
#!/bin/bash
echo "Enter first number "
read x
echo "Enter second number "
read y
((sum=x+y))
echo "The addition of the numbers is $sum"

cdac@LAPTOP-NQM5IGSU:~$ chmod +x addition.sh
cdac@LAPTOP-NQM5IGSU:~$ ./addition.sh
Enter first number
12
Enter second number
23
The addition of the numbers is 35
```

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

```
cdac@LAPTOP-NQM5IGSU:~$ vi evenodd.sh
cdac@LAPTOP-NQM5IGSU:~$ ./evenodd.sh
Enter a number :
3
Odd Number
cdac@LAPTOP-NQM5IGSU:~$ cat evenodd.sh
#!/bin/bash
echo "Enter a number : "
read num
if [ $((num%2)) -eq 0 ]; then
        echo "Even Number"
else
        echo "Odd Number"
fi
```

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

```
cdac@LAPTOP-NQM5IGSU:~$ vi forloop.sh
cdac@LAPTOP-NQM5IGSU:~$ chmod +x forloop.sh
cdac@LAPTOP-NQM5IGSU:~$ ./forloop.sh
1
2
3
4
5
cdac@LAPTOP-NQM5IGSU:~$ cat forloop.sh
#!/bin/bash
for var1 in 1 2 3 4 5
do
        echo "$var1"
done
```

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

```
cdac@LAPTOP-NQM5IGSU:~$ vi whileloop.sh
cdac@LAPTOP-NQM5IGSU:~$ chmod +x whileloop.sh
cdac@LAPTOP-NQM5IGSU:~$ ./whileloop.sh
1
2
3
4
5
cdac@LAPTOP-NQM5IGSU:~$ cat whileloop.sh
#!/bin/bash
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".

```
cdac@LAPTOP-NQM5IGSU:~$ vi checkfile.sh
cdac@LAPTOP-NQM5IGSU:~$ cat checkfile.sh
#!/bin/bash
if [ -f file.txt ]; then
        echo "File exits"
else
        echo "File does not exist"
fi

cdac@LAPTOP-NQM5IGSU:~$ chmod +x checkfile.sh
cdac@LAPTOP-NQM5IGSU:~$ ./checkfile.sh
File exits
```

**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.

```
cdac@LAPTOP-NQM5IGSU:~$ vi numbercheck.sh
cdac@LAPTOP-NQM5IGSU:~$ cat numbercheck.sh
#!/bin/bash
echo -n "enter a number: "
read num
if [ $num -gt 10 ]; then
        echo "The number is greater than 10"
else
        echo "The number is less than 10"
fi

cdac@LAPTOP-NQM5IGSU:~$ ./numbercheck.sh
enter a number: 12
The number is greater than 10
```

**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.

```
cdac@LAPTOP-NQM5IGSU:~$ vi multiplicationtable.sh
cdac@LAPTOP-NQM5IGSU:~$ ./multiplicationtable.sh
   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
   6  12  18  24  30
   7  14  21  28  35
   8  16  24  32  40
   9  18  27  36  45
  10  20  30  40  50
cdac@LAPTOP-NQM5IGSU:~$ cat multiplicationtable.sh
#!/bin/bash
for i in {1..10}
do
        for j in {1..5}
        do
                printf "%4d" $((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.

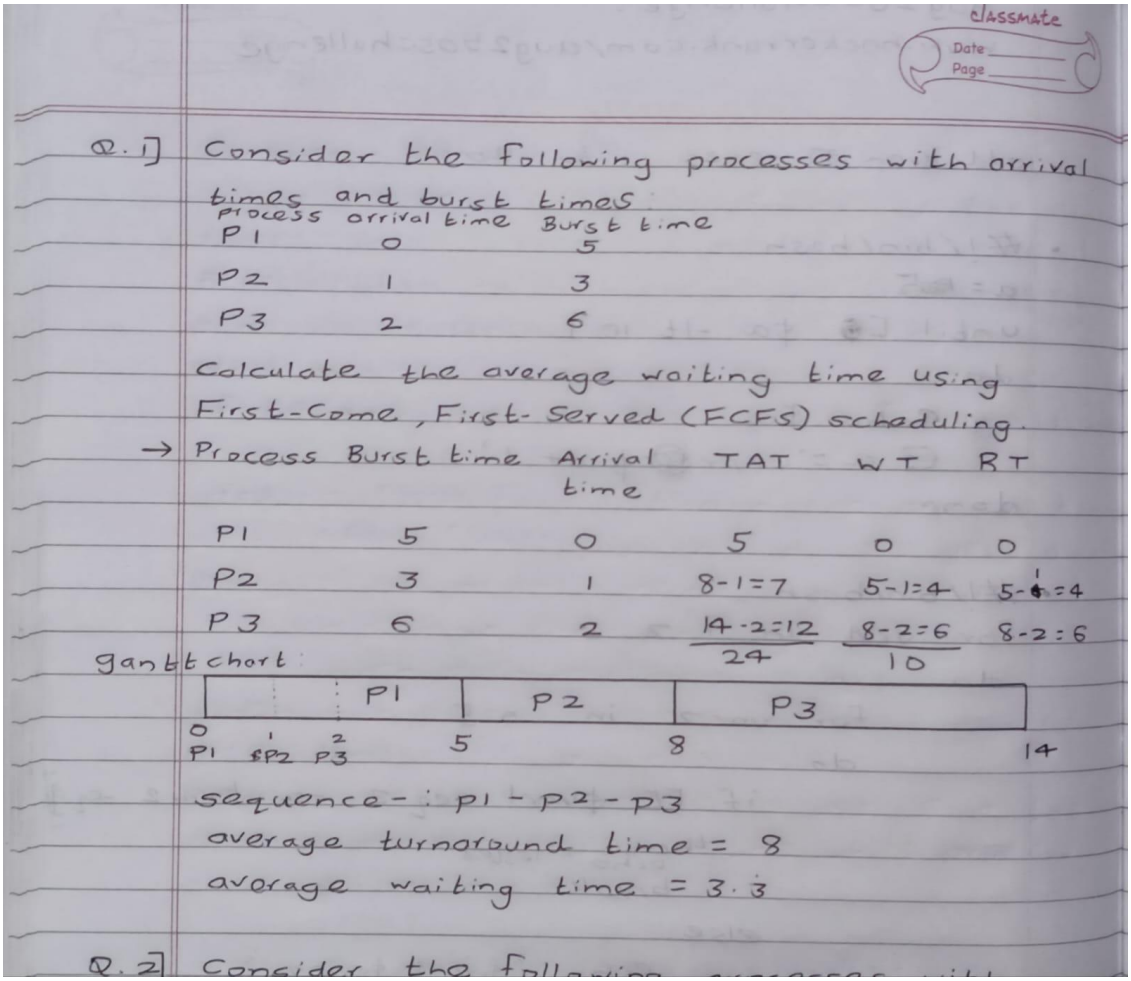
```
cdac@LAPTOP-NQM5IGSU:~$ vi onlypositive.sh
cdac@LAPTOP-NQM5IGSU:~$ cat onlypositive.sh
#!/bin/bash
while true
do
        echo -n "Enter a number (neagative to exist) : "
        read num
        if [ $num -lt 0 ]; then
                echo "Negative number entered.."
                break
        else
                square=$((num*num))
                echo "The square of $num is $square"
        fi
done

cdac@LAPTOP-NQM5IGSU:~$ chmod +x onlypositive.sh
cdac@LAPTOP-NQM5IGSU:~$ ./onlypositive.sh
Enter a number (neagative to exist) : 2
The square of 2 is 4
Enter a number (neagative to exist) : 5
The square of 5 is 25
Enter a number (neagative to exist) : 1
The square of 1 is 1
Enter a number (neagative to exist) : -2
Negative number entered..
```

# 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.




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.

sequence - : P1 - P2 - P3

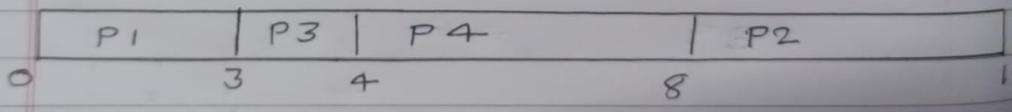
average turnaround time = 8

average waiting time = 3.3

Q.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 | burst time | Arrival time | TAT | WT | RT |
|---|---|---|---|---|---|
| P1 | 3 | 0 | 3 | 0 | 0 |
| P2 | 5 | 1 | 12 | 7 | 7 |
| P3 | 1 | 2 | 2 | 1 | 1 |
| P4 | 4 | 3 | 5 | 1 | 1 |
| | | | 22 | 9 | |

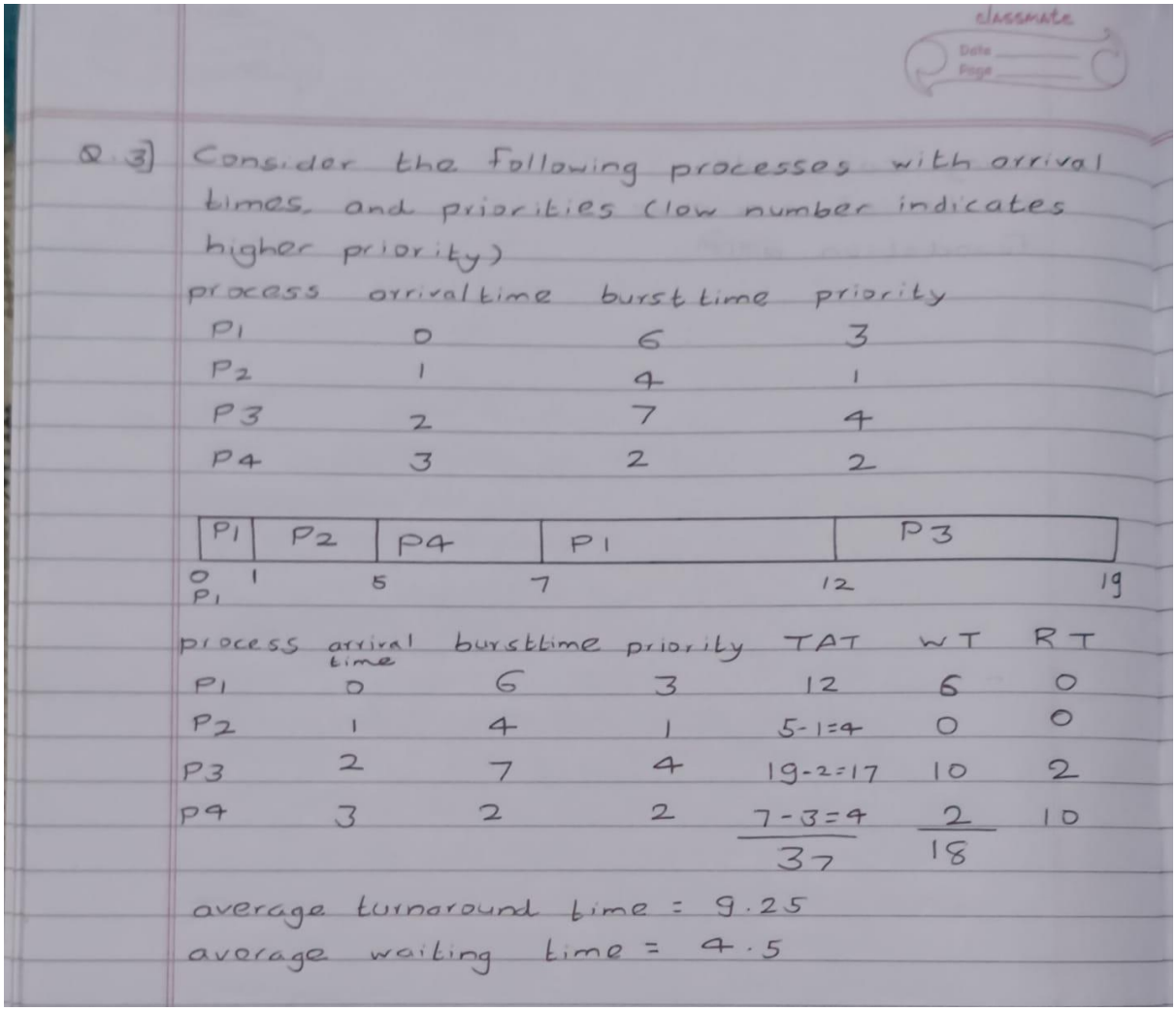
average turnaround time = 5.5

average waiting time = 2.25

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.



Q.3] Consider the following processes with arrival times and priorities (low number indicates higher priority)

| process | arrival time | burst time | priority | TAT | WT | RT |
|---|---|---|---|---|---|---|
| P1 | 0 | 6 | 3 | 12 | 6 | 0 |
| P2 | 1 | 4 | 1 | 5-1=4 | 0 | 0 |
| P3 | 2 | 7 | 4 | 19-2=17 | 10 | 2 |
| P4 | 3 | 2 | 2 | 7-3=4 | 2 | 10 |
| | | | | 37 | 18 | |

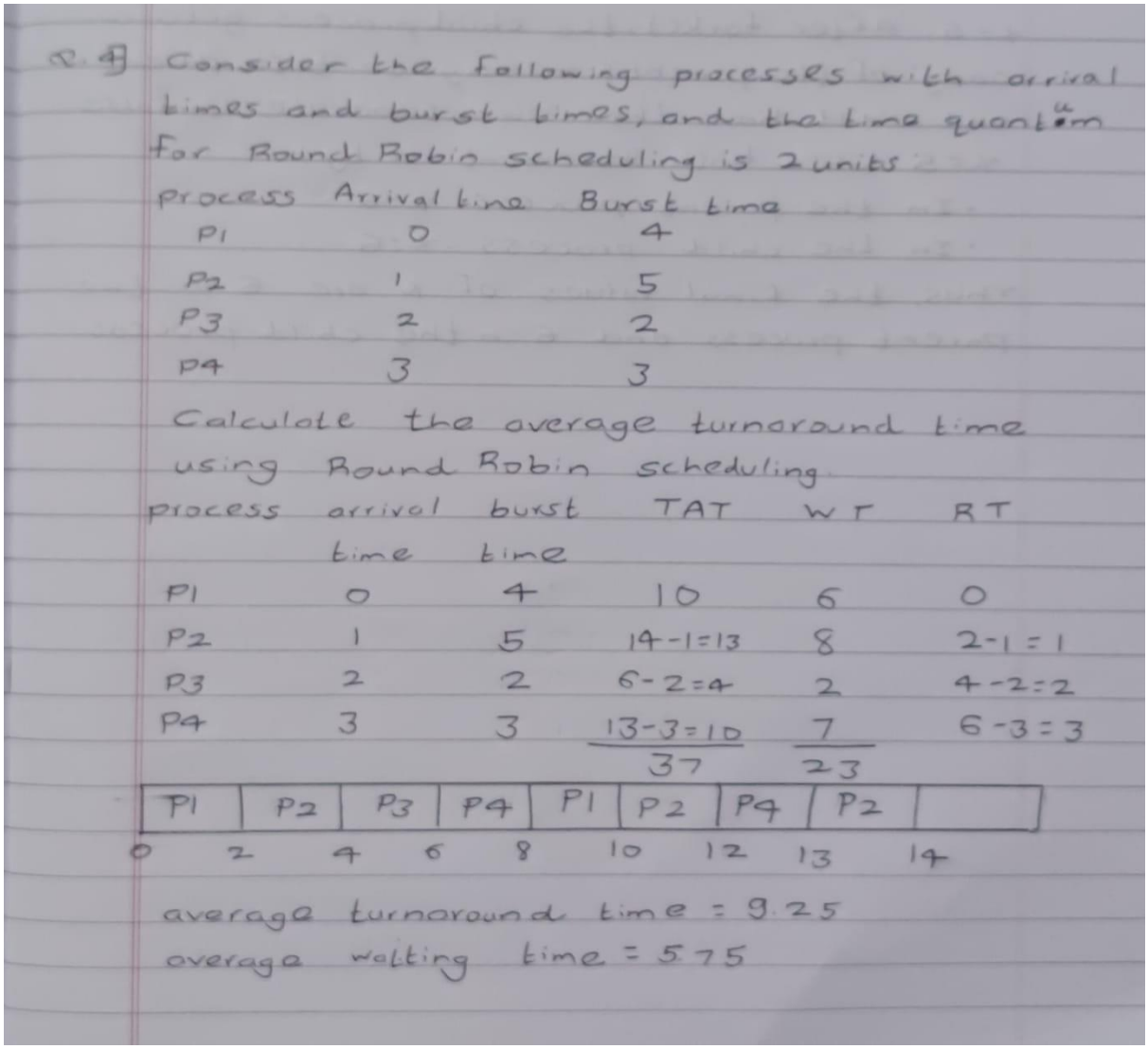


average turnaround time = 9.25
average waiting time = 4.5

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

| 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.



Q.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 | arrival time | burst time | TAT | WT | RT |
|---|---|---|---|---|---|
| P1 | 0 | 4 | 10 | 6 | 0 |
| P2 | 1 | 5 | 14-1=13 | 8 | 2-1=1 |
| P3 | 2 | 2 | 6-2=4 | 2 | 4-2=2 |
| P4 | 3 | 3 | 13-3=10 | 7 | 6-3=3 |
| | | | 37 | 23 | |

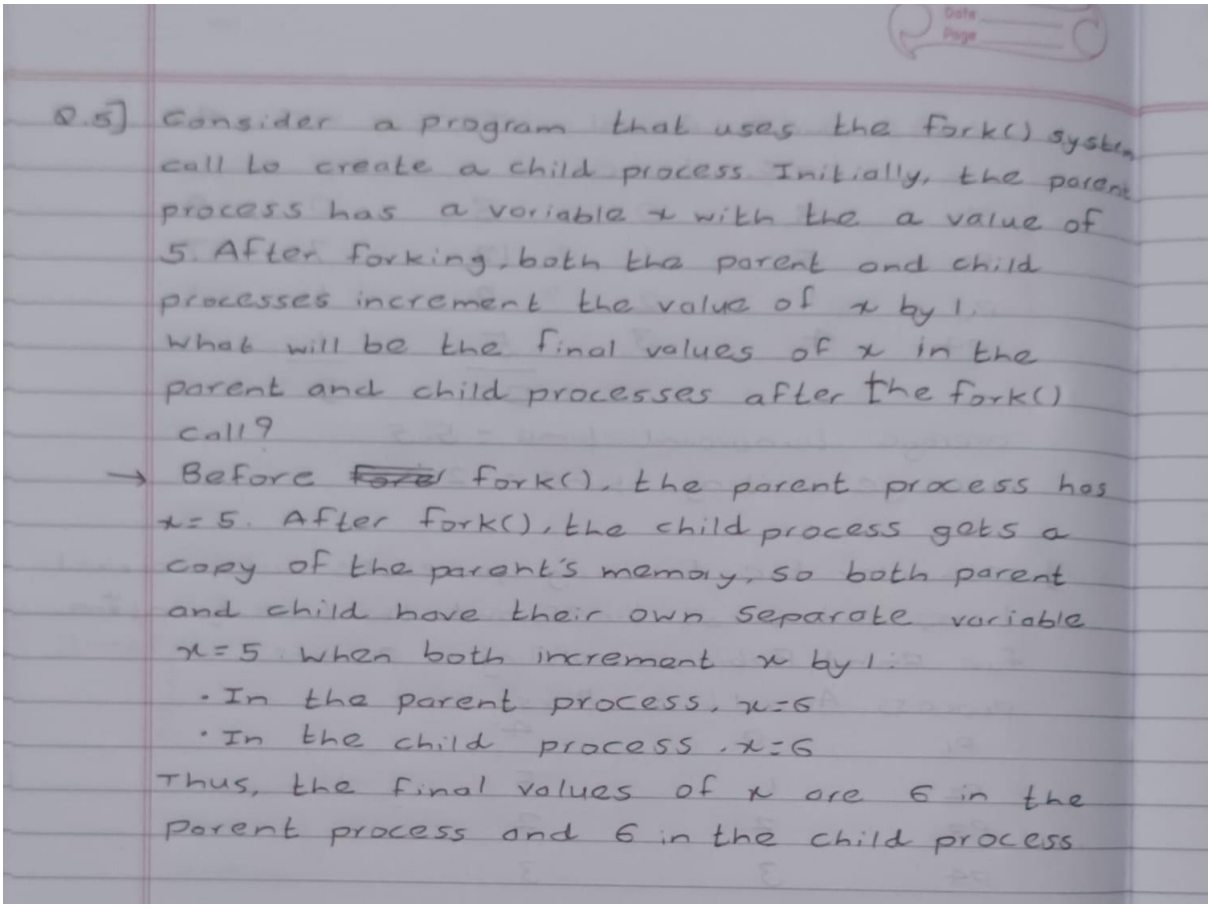
| P1 | P2 | P3 | P4 | P1 | P2 | P4 | P2 | |
|---|---|---|---|---|---|---|---|---|

0 2 4 6 8 10 12 13 14

average turnaround time = 9.25

average waiting time = 5.75

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?



Q.5] Consider a program that uses the fork() system call to create a child process. Initially, the parent process has a variable x with the 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?

→ Before ~~fork~~ fork(), the parent process has x = 5. After fork(), the child process gets a copy of the parent's memory, so both parent and child have their own separate variable x = 5 when both increment x by 1:

- In the parent process, x = 6
- In the child process, x = 6

Thus, the final values of x are 6 in the Parent process and 6 in the child process