Assignment 2

Part A

What will the following commands do?

- echo "Hello, World!" -> Print Hello, World! In bash terminal
- name="Productive"-> sets the value of variable name equal to the string "Productive"
- touch file.txt-> Creates a new empty file named file.txt
- ls -a -> lists all files and directories in current director including the hidden .name files
- rm file.txt->removes/deletes a file
- cp file1.txt file2.txt->copies a file
- mv file.txt /path/to/directory/->moves a file to the specified path
- chmod 755 script.sh->gives user rwx group rx other rx
- grep "pattern" file.txt-> searches the word pattern in file.txt
- kill PID->kills a process whos id is provided by user
- mkdir mydir && cd mydir && touch file.txt && echo "Hello, World!" > file.txt && cat file.txt -> && (logical AND) operator is used here which enables the user to run multiple commands in single command.The above command produces a series of results where output of previous command acts as input for a next command. At first, mkdir command creates a mydir directory in the current directory. cd command is then used to change current directory to new created mydir directory. Touch file.txt creates an empty file named file.txt. Further, echo command will display the message "Hello World" on the terminal. This output of echo command is inserted into file.txt using (>) redirect operator. Finally, contents of file.txt are displayed using cat command.
- ls -l | grep ".txt"-> Lists files in directory with permission specifiers and display .txt type files only
- cat file1.txt file2.txt | sort | uniq -> The above command uses piping to combine the output of cat sort and uniq commands. First command i.e. cat command is used to display the contents of file1.txt followed by contents of file2.txt. sort command is used to perform alphanumeric sort on the result of cat command. Contents of file1.txt and file2.txt are sorted separately in the result. Finally, uniq command is use to display only distinct lines in the result.
- ls -l | grep "^d" -> ls command lists the files and directories in long format. grep "^d" command filters the output to show only lines that start with "d" which in the ls -l output indicates directories.
- grep -r "pattern" /path/to/directory/ -> recursively search for given pattern "pattern" in the user specified directory path "/path/to/directory/"
- cat file1.txt file2.txt | sort | uniq –d -> cat command displays the content of file1.txt followed by file2.txt. sort command is used to perform alphanumeric sort on the result of cat command. Contents of file1.txt and file2.txt are sorted separately in the result. uniq -d command is used to display only duplicate lines in the previous output.
- chmod 644 file.txt ->User rw Group r others r
- cp -r source_directory destination_directory -> used to copy the source_directory and all the sub directories in source_directory as -r (recursively modifier) to destination directory.
- find /path/to/search -name "*.txt" -> searches /path/to/search directory and its subdirectories for any file ending with .txt pattern.
- chmod u+x file.txt ->Add execute permission to user
- echo $PATH -> displays the value of system environment variable that stores directories where executable programs are located.

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 cp is used to copy
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 statement chmod is used
2. **cpy** is used to copy files and directories. Incorrect statement cp is used
3. **mkfile** is used to create a new file. Incorrect statement mkdir is used
4. **catx** is used to concatenate files.Correct Statement
5. **rn** is used to rename files.Incorrect mv is used to rename files/directories

Part C

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

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

Question 2: 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.

```
  ->  read num
      echo $num
```

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

```
  -> #!/bin/bash
     c=$(echo "$1 + $2"|bc)
     echo $c
```

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

```
   ->#!/bin/bash
     if [ $(($1 % 2)) -eq 0 ]; then
        echo "$1 is even"
     else
        echo "$1 is odd"
     fi
```

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

```
     -> #!/bin/bash
        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.

```
     -> #!/bin/bash
          counter=1
          while [ $counter -le 5 ]
          do
          echo $counter
         ((counter++))
         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".

```
     ->#!/bin/bash
        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.

```
->#!/bin/bash
    echo "Please enter a number:"
    read number
    if [ "$number" -gt 10 ]; then
    echo "Number is greater than 10"
   else
   echo "Number is not greater 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.

```
-> #!/bin/bash
   for i in {1..5}; do
   printf "%-2d |" $i
   for j in {1..5}; do
   printf "%-4d" $((i * j))
   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.

```
-> #!/bin/bash
   while true; do
   read -p "Enter a number (negative number to exit): " number
   if [[ $number -lt 0 ]]; then
      echo "Exiting the program."
   break
   else
      square=$((number * number))
      echo "The square of $number is $square"
   fi
   done
```

Part D

Common Interview Questions (Must know)

1. What is an operating system, and what are its primary functions?
2. Explain the difference between process and thread.
3. What is virtual memory, and how does it work?
4. Describe the difference between multiprogramming, multitasking, and multiprocessing.
5. What is a file system, and what are its components?
6. What is a deadlock, and how can it be prevented?
7. Explain the difference between a kernel and a shell.
8. What is CPU scheduling, and why is it important?
9. How does a system call work?
10. What is the purpose of device drivers in an operating system?
11. Explain the role of the page table in virtual memory management.
12. What is thrashing, and how can it be avoided?
13. Describe the concept of a semaphore and its use in synchronization.
14. How does an operating system handle process synchronization?
15. What is the purpose of an interrupt in operating systems?
16. Explain the concept of a file descriptor.
17. How does a system recover from a system crash?
18. Describe the difference between a monolithic kernel and a microkernel.
19. What is the difference between internal and external fragmentation?
20. How does an operating system manage I/O operations?
21. Explain the difference between preemptive and non-preemptive scheduling.
22. What is round-robin scheduling, and how does it work?
23. Describe the priority scheduling algorithm. How is priority assigned to processes?
24. What is the shortest job next (SJN) scheduling algorithm, and when is it used?
25. Explain the concept of multilevel queue scheduling.
26. What is a process control block (PCB), and what information does it contain?
27. Describe the process state diagram and the transitions between different process states.
28. How does a process communicate with another process in an operating system?
29. What is process synchronization, and why is it important?
30. Explain the concept of a zombie process and how it is created.
31. Describe the difference between internal fragmentation and external fragmentation.
32. What is demand paging, and how does it improve memory management efficiency?
33. Explain the role of the page table in virtual memory management.
34. How does a memory management unit (MMU) work?
35. What is thrashing, and how can it be avoided in virtual memory systems?
36. What is a system call, and how does it facilitate communication between user programs and the operating system?
37. Describe the difference between a monolithic kernel and a microkernel.
38. How does an operating system handle I/O operations?
39. Explain the concept of a race condition and how it can be prevented.
40. Describe the role of device drivers in an operating system.
41. What is a zombie process, and how does it occur? How can a zombie process be prevented?
42. Explain the concept of an orphan process. How does an operating system handle orphan processes?
43. What is the relationship between a parent process and a child process in the context of process management?
44. How does the fork() system call work in creating a new process in Unix-like operating systems?
45. Describe how a parent process can wait for a child process to finish execution.
46. What is the significance of the exit status of a child process in the wait() system call?
47. How can a parent process terminate a child process in Unix-like operating systems?

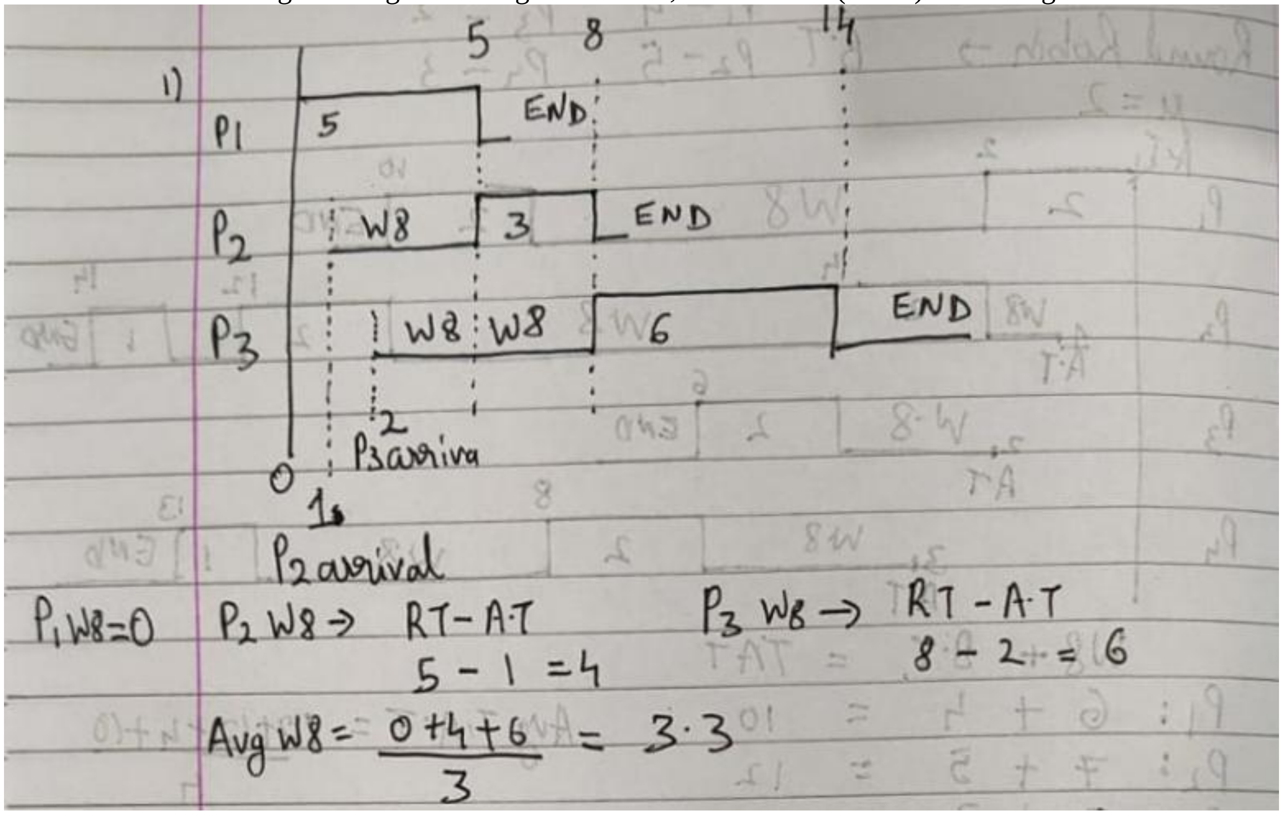
48. Explain the difference between a process group and a session in Unix-like operating systems.
49. Describe how the exec() family of functions is used to replace the current process image with a new one.
50. What is the purpose of the waitpid() system call in process management? How does it differ from wait()?
51. How does process termination occur in Unix-like operating systems?
52. What is the role of the long-term scheduler in the process scheduling hierarchy? How does it influence the degree of multiprogramming in an operating system?
53. How does the short-term scheduler differ from the long-term and medium-term schedulers in terms of frequency of execution and the scope of its decisions?
54. Describe a scenario where the medium-term scheduler would be invoked and explain how it helps manage system resources more efficiently.

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.



P_1 W8 = 0 P_2 W8 → RT − A·T P_3 W8 → RT − A·T

5 − 1 = 4 8 − 2 = 6

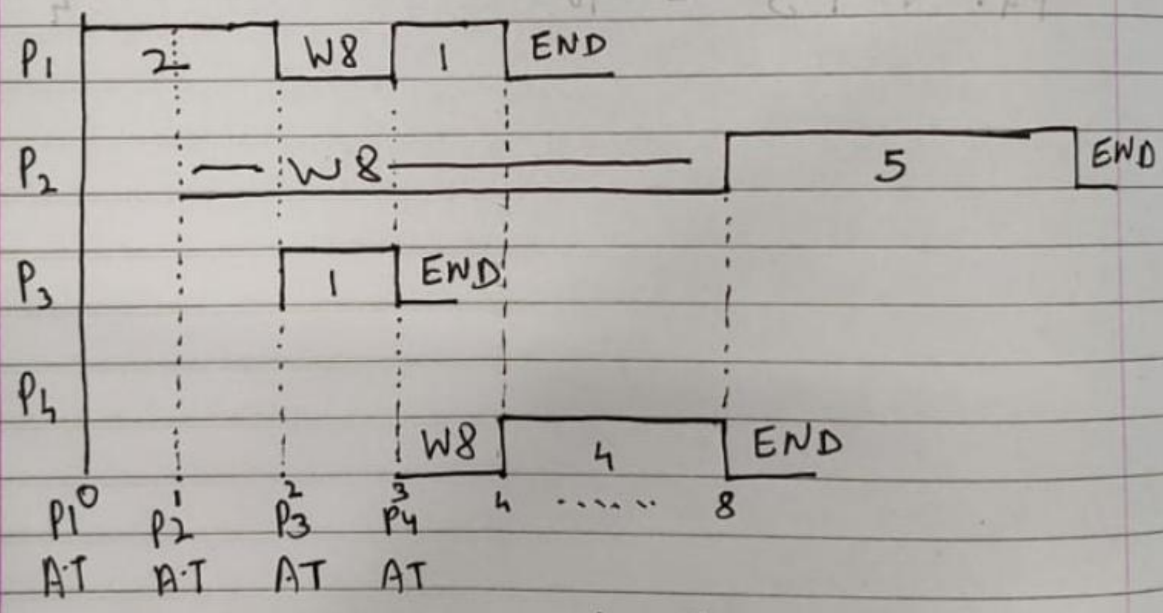
Avg W8 = $\frac{0+4+6}{3}$ = 3·3

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.

2) SJF Preemptive

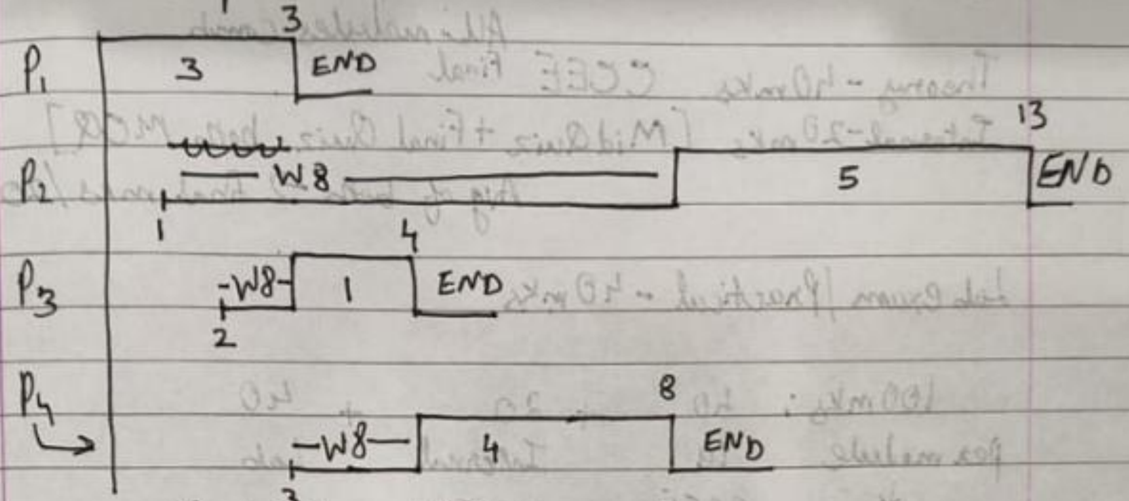


	W8		BT		TAT
$P_1 \rightarrow$	1	+	3	=	4
$P_2 \rightarrow$	7	+	5	=	12
$P_3 \rightarrow$	0	+	1	=	1
$P_4 \rightarrow$	1	+	4	=	5

Avg TAT preemptive $\rightarrow$

$$\frac{4+12+1+5}{4} = \frac{22}{4} = 5.5$$

2. Non Preemptive



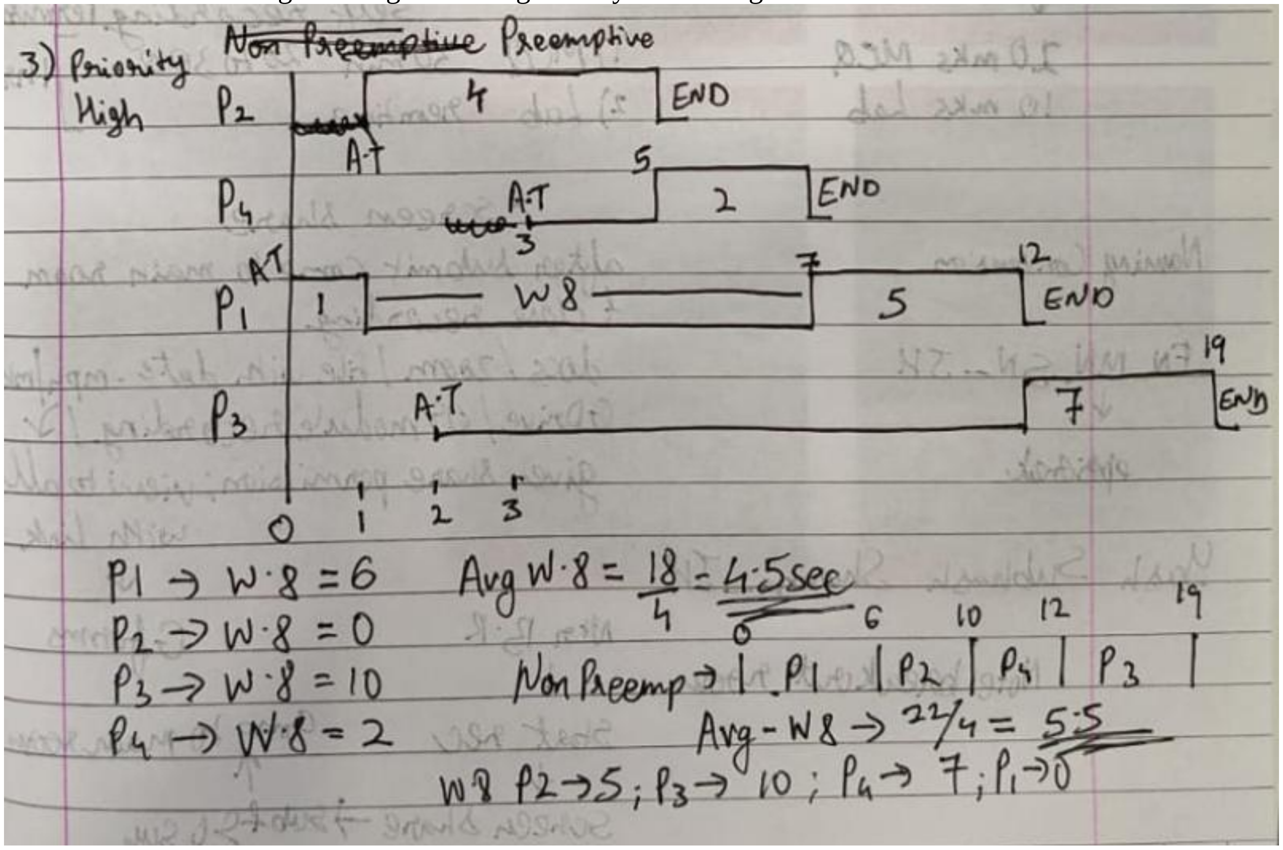
	W8	+	BT		TAT
$P_1 \rightarrow$	0	+	3	=	3
$P_2 \rightarrow$	7	+	5	=	12
$P_3 \rightarrow$	1	+	1	=	2
$P_4 \rightarrow$	1	+	4	=	5

$$\text{Avg TAT} = \frac{3+12+2+5}{4} = \frac{22}{4} = 5.5$$

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.



P1 → W.8 = 6
P2 → W.8 = 0
P3 → W.8 = 10
P4 → W.8 = 2

Avg W.8 = $\frac{18}{4}$ = 4.5sec

0 6 10 12 19
Non Preemp → | P1 | P2 | P4 | P3 |

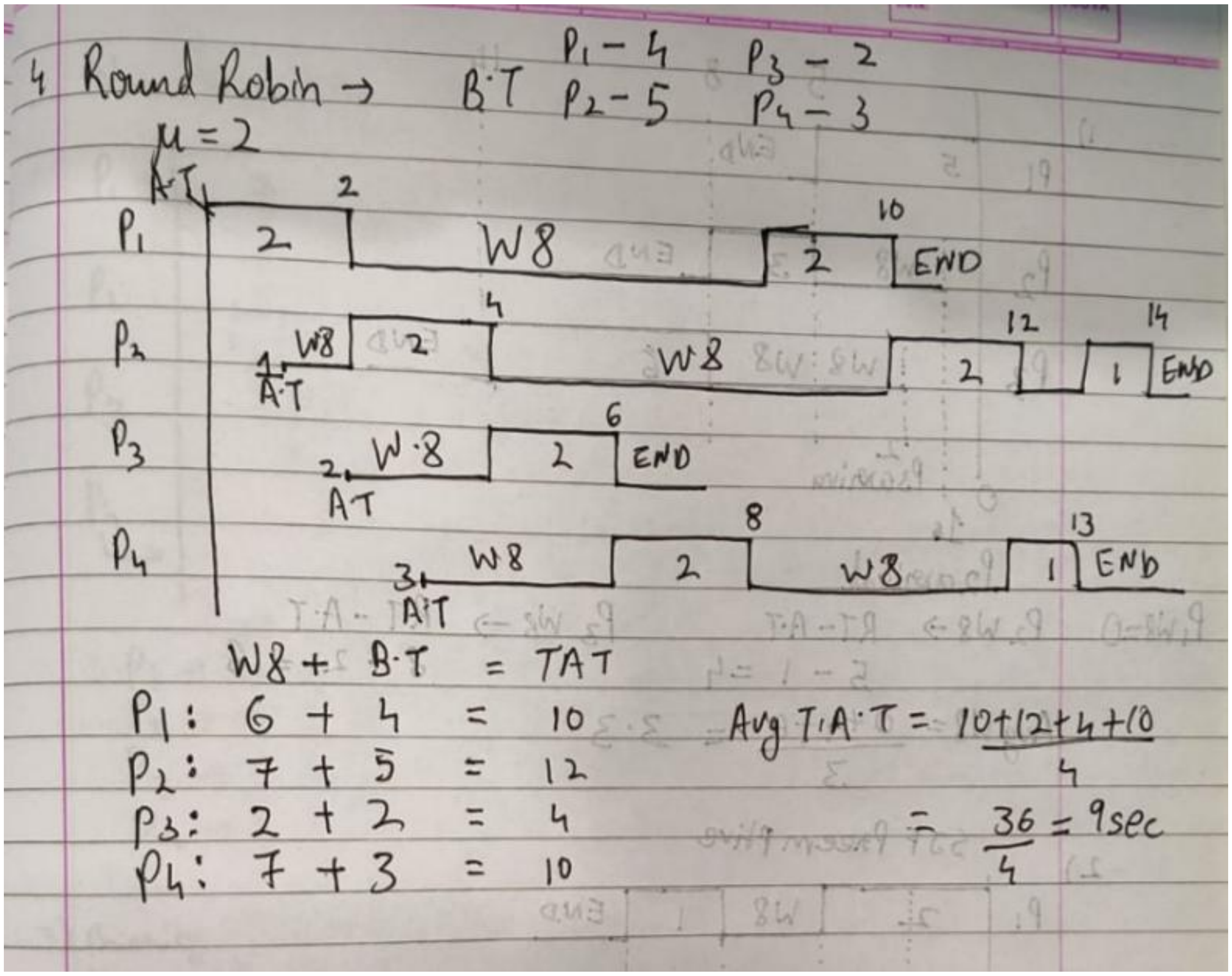
Avg - W8 → 22/4 = 5.5

W8 P2→5; P3→10; P4→7; P1→0

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.


4 Round Robin → B.T $P_1 - 4$ $P_3 - 2$

$P_2 - 5$ $P_4 - 3$

$\mu = 2$

W8 + B.T	= TAT
P_1: 6 + 4	= 10
P_2: 7 + 5	= 12
P_3: 2 + 2	= 4
P_4: 7 + 3	= 10

$$\text{Avg T.A.T} = \frac{10+12+4+10}{4} = \frac{36}{4} = 9\text{sec}$$

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?

o When the fork() system call is used, it creates a child process that has its own copy of the parent's memory.

o Before forking, the parent has a variable x = 5. After the fork, both the parent and child have separate copies of x, still equal to 5.

o Each process then increments x by 1, so both the parent and child have x = 6, but in their own separate memory.

o In parent process, x=6. In child process, x=6

Submission Guidelines:

- Document each step of your solution and any challenges faced.
- Upload it on your GitHub repository

Additional Tips:

- Experiment with different options and parameters of each command to explore their functionalities.
- This assignment is tailored to align with interview expectations, CCEE standards, and industry demands.
- If you complete this then your preparation will be skyrocketed