Part A

What will the following commands do?

* •  echo "Hello, World!"

It will Print Hello,World! To console

* •  name="Productive"

It will assign Productive as string to variable name

* •  touch file.txt

It will create an empty file.txt text file in the current directory

* •  ls-a

It will list all files and folders in current directory

* •  rm file.txt

It will delete file.txt file

* •  cp file1.txt file2.txt

It will copy contents of file1.txt to file2.txt

* •  mv file.txt /path/to/directory/

It will move file.txt to path specified

* •  chmod 755 script.sh

It will grant all permission to user, read and execute permissions to group and others for script.sh

* •  grep "pattern" file.txt

It will match the pattern specified with the contents of file.txt and return the lines matched

* •  kill PID

It will terminate the process

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

It will first create mydir directory and move into it  
Then create an empty file file.txt and using redirection operator it will write hello world in file.txt.Then at last it will print contents of file to console i.e Hello, World!

* •  ls -l | grep ".txt"

It will list all text files (long listing)

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

It will display unique lines from file1.txt and file2.txt

* •  ls -l | grep "^d"

It will list all the directories since we need to search for lines starting with d

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

It will recursively search for the specified pattern in the current and sub directories

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

It will display unique lines sorted from file1.txt and file2.txt

* •  chmod 644 file.txt

Assigns read and write permission to user, and only read permission to other and group for file.txt

* •  cp -r source\_directory destination\_directory

Copy all files and directories recursively in current and subdirectories to destination directory

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

It will display all .txt file present at the path specified

* •  chmod u+x file.txt

Assigns execute permission to user for file.txt

* •  echo $PATH

It will display path to bin where executable files are stored

1. Part B
2. 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. Correct command: chmod
2. cpy is used to copy files and directories. Correct command:cp
3. mkfile is used to create a new file. Correct command:mkdir
4. catx is used to concatenate files. Correct command:cat
5. rn is used to rename files. Correct command:mv

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.

#!/bin/bash

name="CDAC Mumbai"  
echo "$name"

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  
echo "The number is $number"

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

#!/bin/bash

echo "Enter number"

read number1

read number2

((sum=$number1+$number2))

echo "The sum is $sum"

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 number1

((ans = number1 % 2))

if (( number1 % 2 == 0))

then

echo "even"

else

echo "odd"  
fi

Question 6: Write a shell script that uses a for loop to print numbers from 1 to 5. Question 7: Write a shell script that uses a while loop to print numbers from 1 to 5.

#!/bin/bash

for i in {1..5}

do

echo $i  
done

#!/bin/bash

i=1

for ((i=1;i<=5;i=i+1))

do

echo $i

done

#!/bin/bash

i=1

while ((i<=5))

do

echo $i

((i=$i+1))

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

s=$(ls file.txt)

if [ $s = "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

read num

if [ $num -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.

#!/bin/bash

for i in {1..10}; do

for j in {1..5}; do

result=$((i \* j))

printf "%-4d" $result

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.

#!/bin/bash

echo "Enter positive numbers (enter a negative number to exit):"

while true; do

read -p "Enter a number: " num

if [ $num -lt 0 ]; then

echo "Exiting the loop. Negative number entered."

break

fi

square=$((num \* num))

echo "Square of $num 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.

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.

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 |

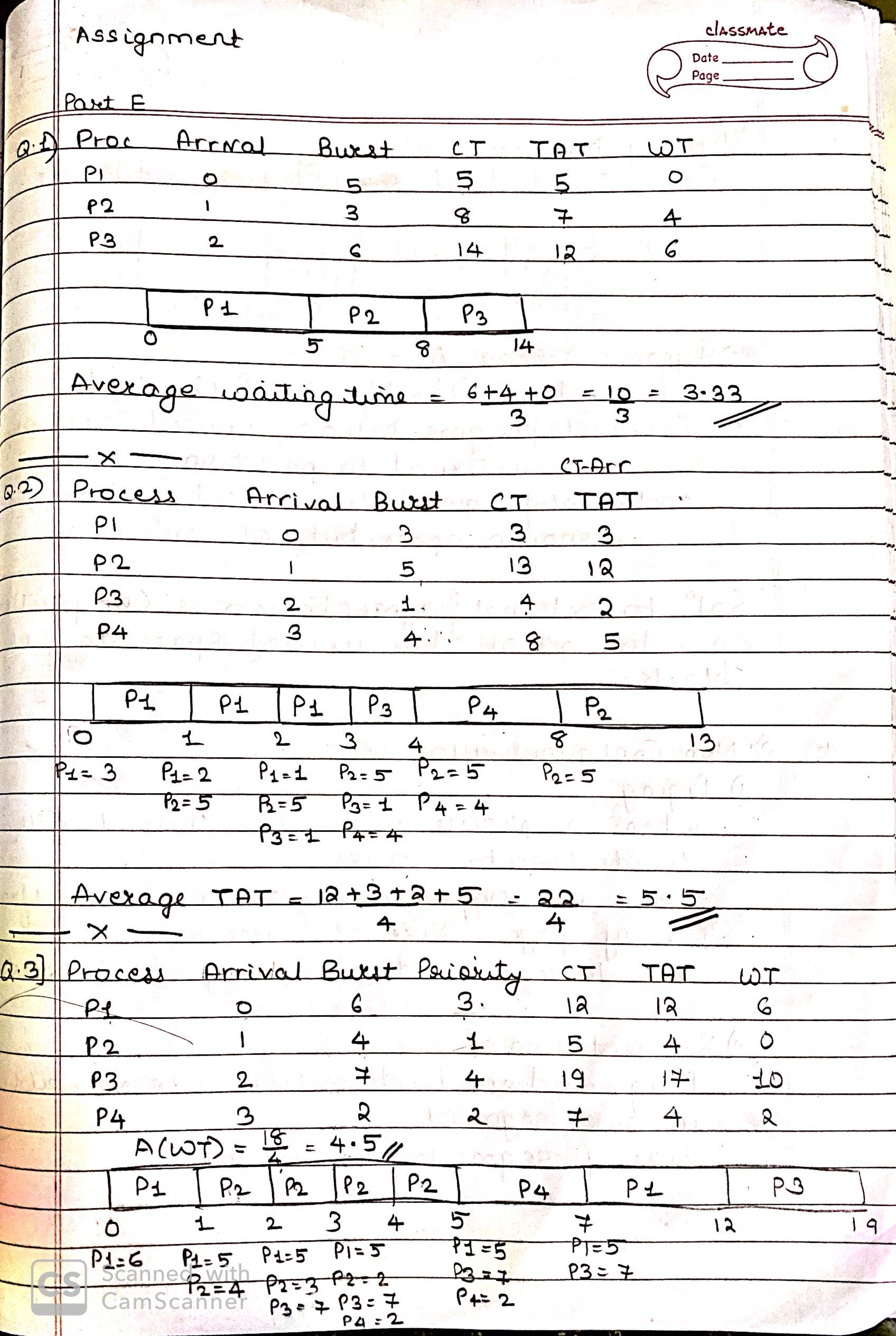
|P3 |2 |7 |4 |  
|P4 |3 |2 |2 |  
Calculate the average waiting time using Priority Scheduling.

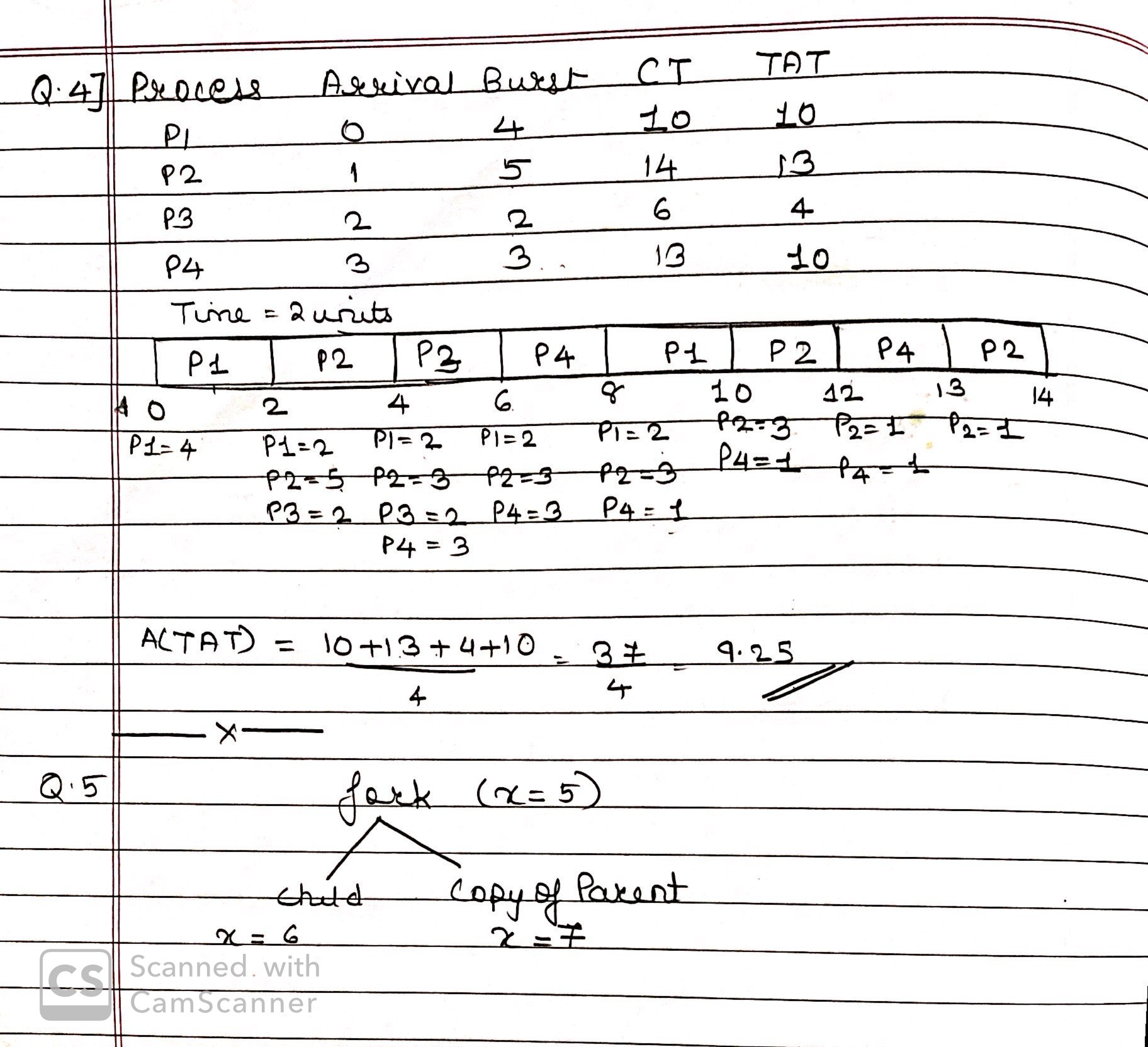
4. Consider the following processes with arrival times and burst times, and the time quantum for Round Robin scheduling is 2 units:

| 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.

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?





Q5) Parent process has x = 5.

After the fork() call, both the parent and child processes have their own separate copies of x= 5.

In both processes, x is incremented by 1. Therefore, in both the parent and child processes, x = 6.

So, the final value of x in both the parent and child processes after the fork() call will be 6.