CDAC MUMBAI

Concepts of Operating System

Assignment 2

Part A

What will the following commands do?

- echo "Hello, World!"

Answer:- This echo command will print Hello,World!.

- name="Productive"

Answer:- This command will assign the string “Productive” to the name variable.

- touch file.txt

Answer:- touch command creates an empty text file.

- ls -a

Answer :- this command will display all files including hidden ones.

- rm file.txt

Answer :- this command is used to delete a file from the directory.

- cp file1.txt file2.txt

Answer :- this command copies the content of file1 into the file2.

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

Answer:- this command moves the file from the current location to the specified directory. If the same file name exist into the output directory it will be overwritten.

- chmod 755 script.sh

Answer:- this command will change the ownership of file 7 which is r-w-x for owner, 5 which is r-x for group, 5 which is r-x for others.

- grep "pattern" file.txt

Answer:- this command is used to find/search the specific word in that file in this case “pattern”.

- kill PID

Answer:- this command is use to terminate a process having a Process Id.

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

Answer:- this command will create directory mydir and move into that dir with cd with touch creates an empty file and with echo hello world will be saved into file.txt and cat command will display the content of that file which is Hello World!

- ls -l | grep ".txt"

Answer:- this command will list all files and their information but filters only gives .txt file

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

Answer:- this command will perform 3 operations 1st will concatenate contents of both files with 2nd sorts the data alphabetically and uniq will print out only uniq entries no duplicates.

- ls -l | grep "^d"

Answer:- this command will list all files and their information and filter line which starts with letter ‘d’.

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

Answer:- this command searches for the string “pattern” searches recursively from the given path of that directory.

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

Answer:- this command concatenates/combines both files and sorts the data alphabetically and displays only duplicate lines.

- chmod 644 file.txt

Answer:- this command will change the mode of the file to 6 that is read and write rw- for owner,4 that is only read r—for group and 4 that is only read r—for others.

- cp -r source_directory destination_directory

Answer:- this command will copy the entire directory including all files and sub directories from one location to other.

• find /path/to/search -name "*.txt"

Answer:- this command is use for searching all files ending with .txt within that specific directory

• chmod u+x file.txt

Answer:- this command will add execute permission to the owner of that file so he will be able to execute that file.

• echo $PATH

Answer:- this command is use to display all environment variables path.

Part B

1. **ls** is used to list files and directories in a directory.

   Answer:- True

2. **mv** is used to move files and directories.

   Answer:- True

3. **cd** is used to copy files and directories.

   Answer:- False

4. **pwd** stands for "print working directory" and displays the current directory.

   Answer:- True

5. **grep** is used to search for patterns in files.

   Answer:- True

6. **chmod 755 file.txt** gives read, write, and execute permissions to the owner, and read and execute permissions to group and others.

   Answer:- True

7. **mkdir -p directory1/directory2** creates nested directories, creating directory2 inside directory1 if directory1 does not exist.

   Answer:- True

8. **rm -rf file.txt** deletes a file forcefully without confirmation.

   Answer:- True.

Identify the Incorrect Commands:

1. **chmodx** is used to change file permissions.

Answer:- Incorrect

2. **cpy** is used to copy files and directories.

Answer:- Incorrect

3. **mkfile** is used to create a new file.

Answer:- Incorrect

4. **catx** is used to concatenate files.

Answer:- Incorrect

5. **rn** is used to rename files.

Answer:- Incorrect

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.

```
cdac@Maverick:~$ cat q3.sh
#!/bin/bash
echo "Enter a number :"
read n
echo "The entered number is $n"
cdac@Maverick:~$ chmod -x q3.sh
```

```
cdac@Maverick:~$ ./q3.sh
-bash: ./q3.sh: Permission denied
cdac@Maverick:~$ cat q3.sh
#!/bin/bash
echo "Enter a number :"
read n
echo "The entered number is $n"
cdac@Maverick:~$ chmod +x q3.sh
cdac@Maverick:~$ ./q3.sh
Enter a number :
9
The entered number is 9
cdac@Maverick:~$
```

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

```
cdac@Maverick:~$ cat s8.sh
#!/bin/bash
echo -n "Enter first number"
read x
echo -n "Enter second number"
read y
(( sum= x+y ))
echo "the result is = $sum"

cdac@Maverick:~$ ./s8.sh
Enter first number3
Enter second number5
the result is = 8
cdac@Maverick:~$
```

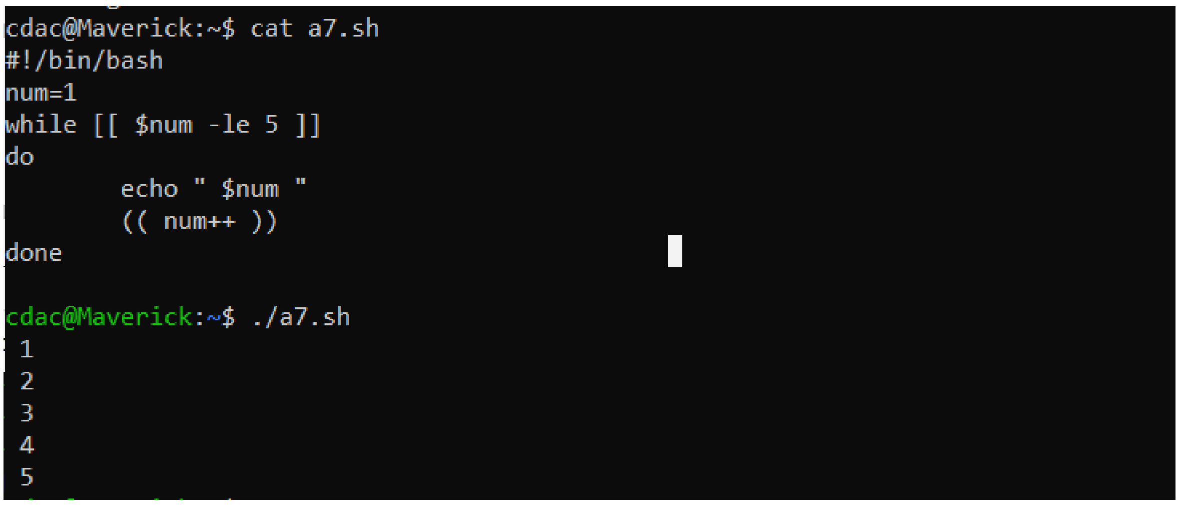
Question 5: Write a shell script that takes a number as input and prints "Even" if it is even, otherwise prints "Odd".

```
cdac@Maverick:~$ vi q4.sh
cdac@Maverick:~$ cat q4.sh
#!/bin/bash
echo "Enter a number :"
read n
if (( n % 2 == 0 )); then
	echo "The given number $n is even"
else
	echo "The given number $n is odd "
fi
cdac@Maverick:~$ ./q4.sh
Enter a number :
7
The given number 7 is odd
cdac@Maverick:~$
```

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

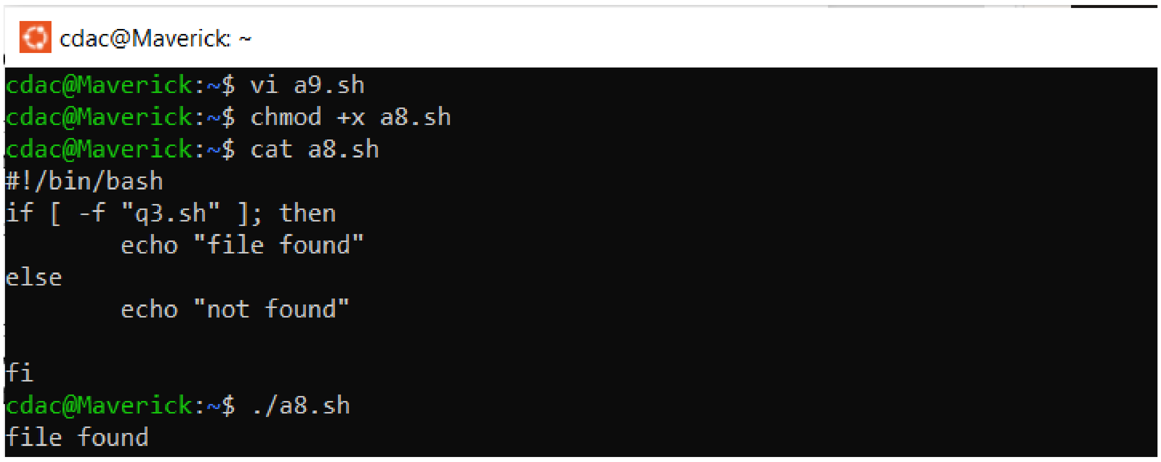
```
cdac@Maverick:~$ cat q5.sh
#!/bin/bash
for i in {1..5}
do
	echo "$i"
done
cdac@Maverick:~$ ./q5.sh
1
2
3
4
5
cdac@Maverick:~$
```

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



```
cdac@Maverick:~$ cat a7.sh
#!/bin/bash
num=1
while [[ $num -le 5 ]]
do
        echo " $num "
        (( num++ ))
done

cdac@Maverick:~$ ./a7.sh
 1
 2
 3
 4
 5
```

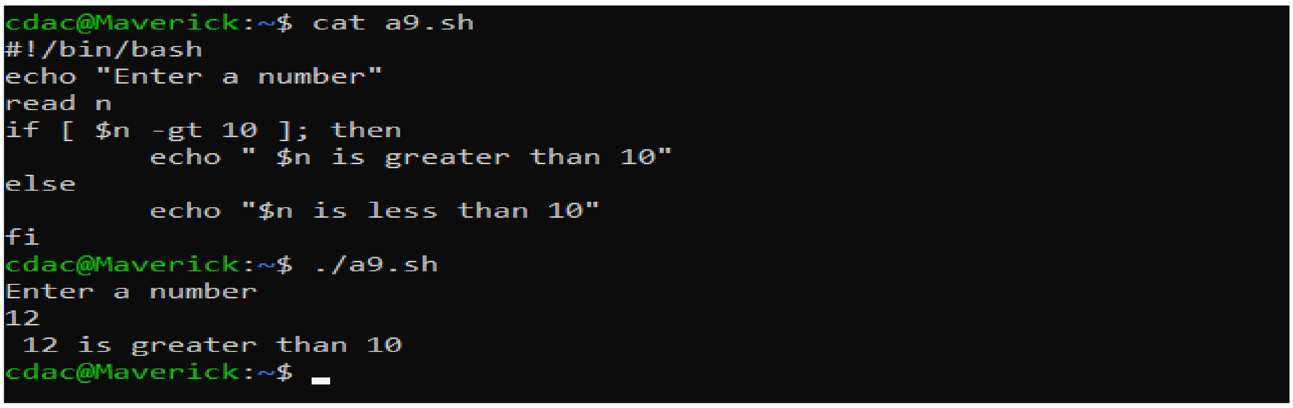
Question 8: Write a shell script that checks if a file named "file.txt" exists in the current directory. If it does, print "File exists", otherwise, print "File does not exist".



cdac@Maverick: ~

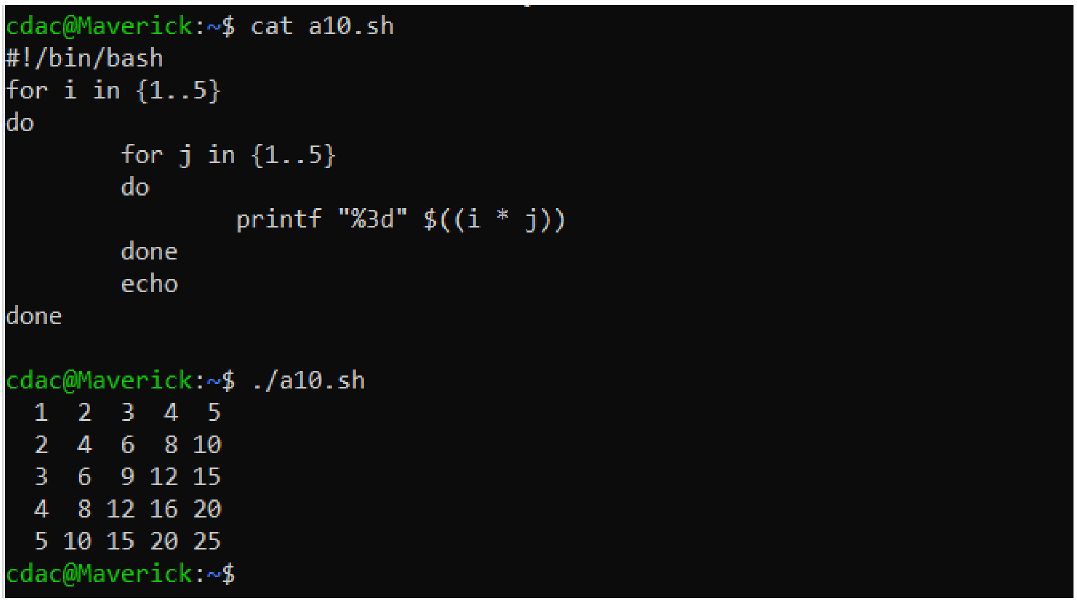
```
cdac@Maverick:~$ vi a9.sh
cdac@Maverick:~$ chmod +x a8.sh
cdac@Maverick:~$ cat a8.sh
#!/bin/bash
if [ -f "q3.sh" ]; then
        echo "file found"
else
        echo "not found"

fi
cdac@Maverick:~$ ./a8.sh
file found
```

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.



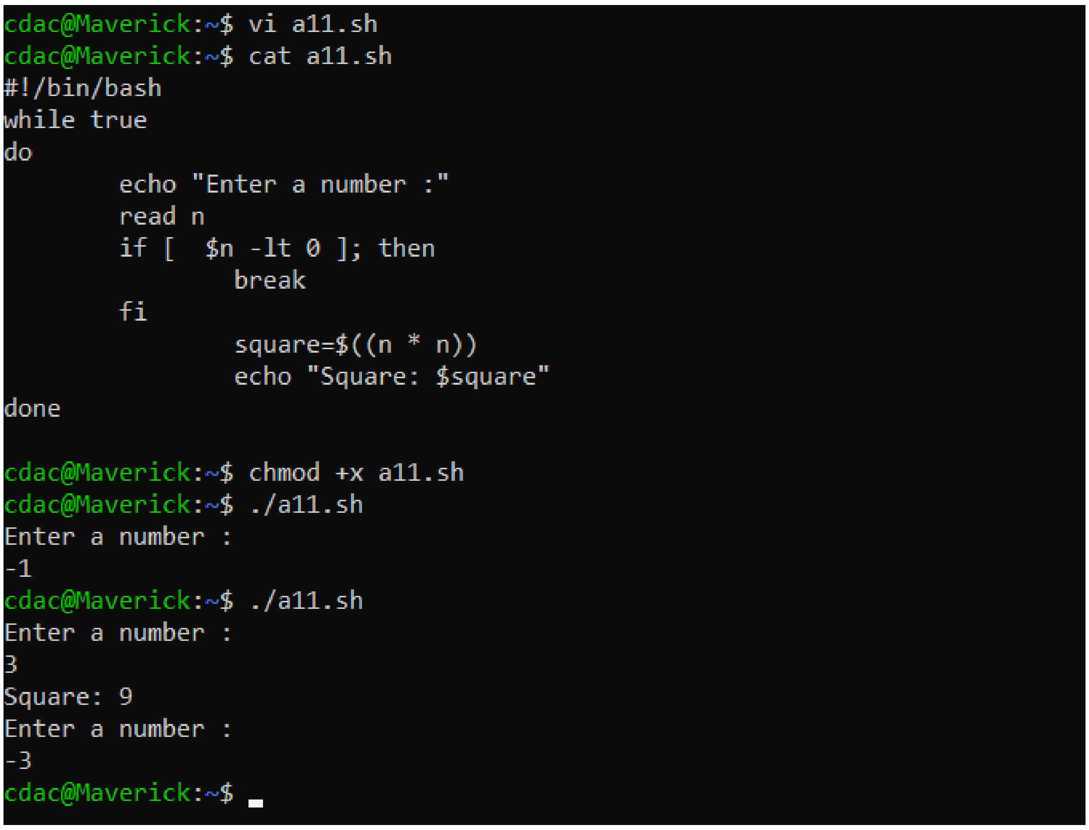
```
cdac@Maverick:~$ cat a9.sh
#!/bin/bash
echo "Enter a number"
read n
if [ $n -gt 10 ]; then
        echo " $n is greater than 10"
else
        echo "$n is less than 10"
fi
cdac@Maverick:~$ ./a9.sh
Enter a number
12
 12 is greater than 10
cdac@Maverick:~$
```

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.



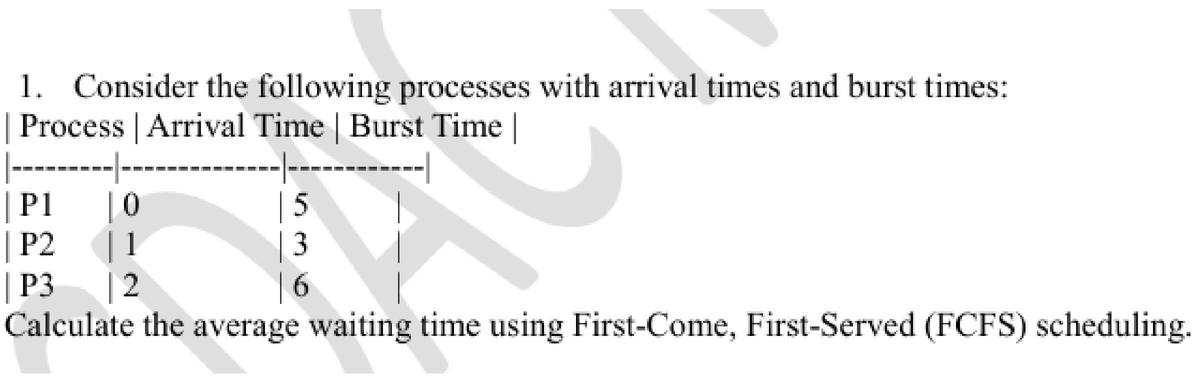
```
cdac@Maverick:~$ cat a10.sh
#!/bin/bash
for i in {1..5}
do
        for j in {1..5}
        do
                printf "%3d" $((i * j))
        done
        echo
done

cdac@Maverick:~$ ./a10.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
cdac@Maverick:~$
```

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@Maverick:~$ vi a11.sh
cdac@Maverick:~$ cat a11.sh
#!/bin/bash
while true
do
        echo "Enter a number :"
        read n
        if [  $n -lt 0 ]; then
                break
        fi
                square=$((n * n))
                echo "Square: $square"
done

cdac@Maverick:~$ chmod +x a11.sh
cdac@Maverick:~$ ./a11.sh
Enter a number :
-1
cdac@Maverick:~$ ./a11.sh
Enter a number :
3
Square: 9
Enter a number :
-3
cdac@Maverick:~$
```

Part E



1. Consider the following processes with arrival times and burst times:

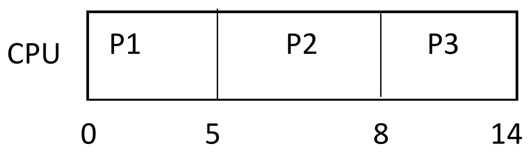
Process	Arrival Time	Burst Time
P1	0	5
P2	1	3
P3	2	6

Calculate the average waiting time using First-Come, First-Served (FCFS) scheduling.

Answer:-

Process	Arrival Time(AT)	Burst Time (BT)	Completion Time (CT)	Turn Around Time(CT-AT)	Waiting Time (WT)
P1	0	5	5	5	0
P2	1	3	8	7	4
P3	2	6	14	12	6
				24	10

Gantt Chart :-

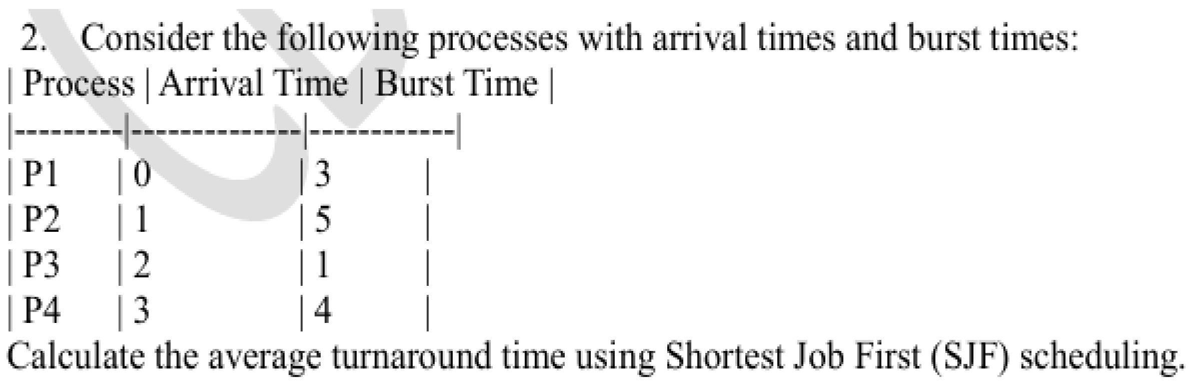


Ready Queue:- P1,P2,P3

Average Waiting Time = Total Waiting Time / Number Of Process

= 24 / 3

Average Waiting Time = 3.33 Units



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	Arrival Time(AT)	Burst Time (BT)	Completion Time (CT)	Turn Around Time(CT-AT)
P1	0	3	3	3
P2	1	5	13	12
P3	2	1	4	2
P4	3	4	8	5
				22

Gantt Chart :- Ready Queue :- P1,P3,P4,P2

P1	P3	P4	P2

0 3 4 8 13

Average Turn Around Time = Total Turn Around Time / Number Of Process

= 22/4

Average Turn Around Time = 5.5 Units

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

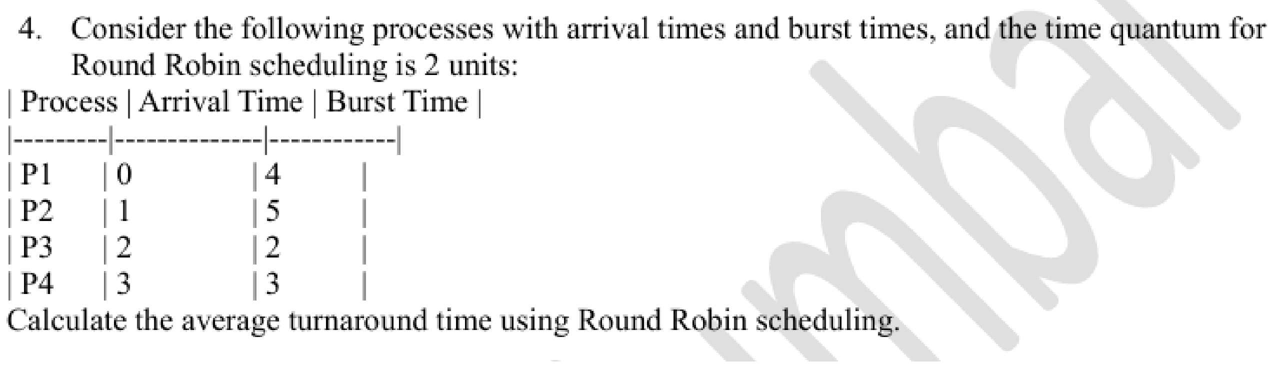
Calculate the average waiting time using Priority Scheduling.

Process	Arrival Time(AT)	Burst Time (BT)	Priority	Completion Time (CT)	Waiting Time (WT)
P1	0	6	3	6	0
P2	1	4	1	10	5
P3	2	7	4	19	7
P4	3	2	2	12	10
					22

Average Waiting Time = Total Waiting Time / Number Of Process

= 22 / 4

Average Waiting Time = 5.5 Units



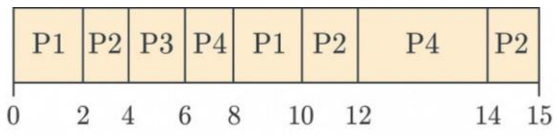
4. Consider the following processes with arrival times and burst times, and the time quantum for Round Robin scheduling is 2 units:

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(AT)	Burst Time (BT)	Completion Time (CT)	Turn Around Time(CT-AT)
P1	0	4	10	10
P2	1	5	15	14
P3	2	2	6	4
P4	3	3	14	11
				39

Gantt Chart : -



Average Turn Around Time = Total Turn Around Time / Number Of Process

= 39/4

Average Turn Around Time = 9.75 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?

Answer:-

- When a program calls a fork(), it creates a new child process that is a copy of the parent.
- This includes the copy of the parents memory space.
- Before fork() x = 5
  But after fork() x increments by 1

- Child increments x = 6
- Parent increments x = 6