NATIONAL TEXTILE UNIVERSITY

DEPARTMENT OF COMPUTER SCIENC

SUBMITTED BY:

Eman Faisal 23-NTU-CS-1149

SECTION SE: 5th (A)

Operating Systems- Assignment1

SUBMITTED TO:

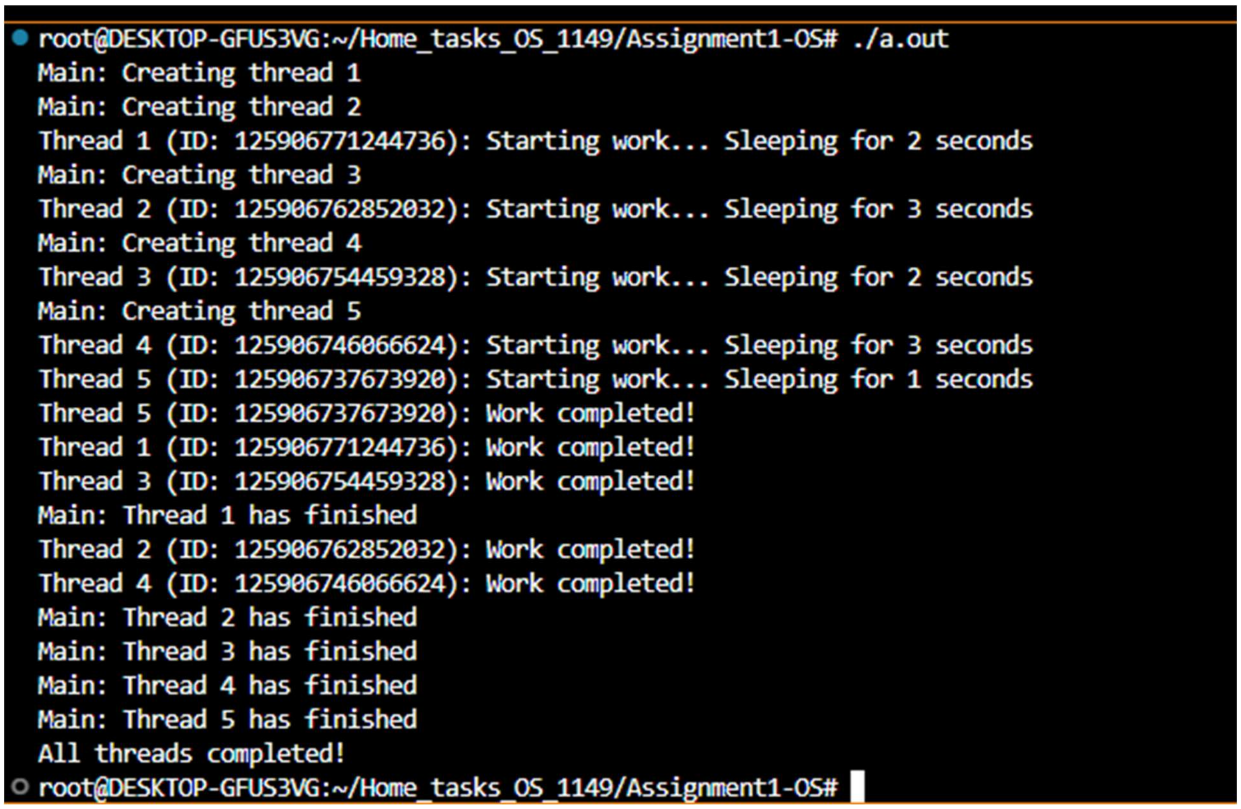
Sir Nasir Mahmood

SUBMISSION DATE: 10/26/25

TASK 1:

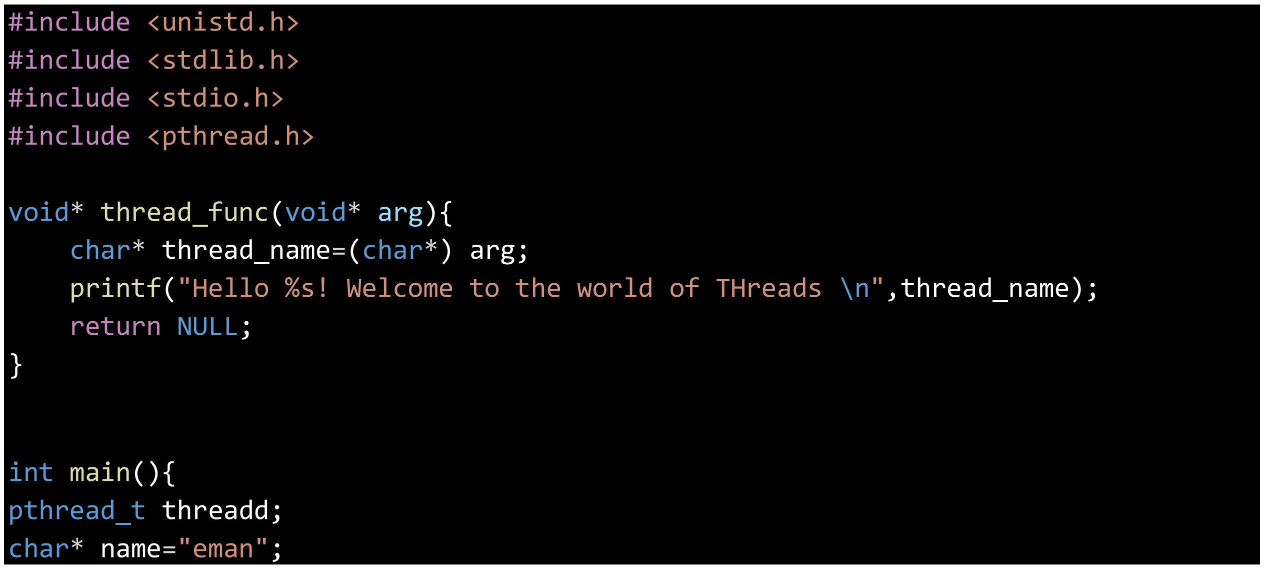
```c
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#include <stdlib.h>
#include <time.h>

void* worker_thread(void* arg) {
    int thread_num = *(int*)arg;
    pthread_t tid = pthread_self();

    int sleep_time = rand() % 3 + 1;

    printf("Thread %d (ID: %lu): Starting work... Sleeping for %d seconds\n",
           thread_num, tid, sleep_time);

    sleep(sleep_time);

    printf("Thread %d (ID: %lu): Work completed!\n", thread_num, tid);
    return NULL;
}

int main() {
    pthread_t threads[5];
    int thread_args[5];
    srand(time(NULL));

    // Create 5 threads
    for (int i = 0; i < 5; i++) {
        thread_args[i] = i + 1;
        printf("Main: Creating thread %d\n", i + 1);
        pthread_create(&threads[i], NULL, worker_thread, &thread_args[i]);
    }
    for (int i = 0; i < 5; i++) {
        pthread_join(threads[i], NULL);
        printf("Main: Thread %d has finished\n", i + 1);
    }

    printf("All threads completed!\n");
    return 0;
}
```

OUTPUT:



```
root@DESKTOP-GFUS3VG:~/Home_tasks_OS_1149/Assignment1-OS# ./a.out
Main: Creating thread 1
Main: Creating thread 2
Thread 1 (ID: 125906771244736): Starting work... Sleeping for 2 seconds
Main: Creating thread 3
Thread 2 (ID: 125906762852032): Starting work... Sleeping for 3 seconds
Main: Creating thread 4
Thread 3 (ID: 125906754459328): Starting work... Sleeping for 2 seconds
Main: Creating thread 5
Thread 4 (ID: 125906746066624): Starting work... Sleeping for 3 seconds
Thread 5 (ID: 125906737673920): Starting work... Sleeping for 1 seconds
Thread 5 (ID: 125906737673920): Work completed!
Thread 1 (ID: 125906771244736): Work completed!
Thread 3 (ID: 125906754459328): Work completed!
Main: Thread 1 has finished
Thread 2 (ID: 125906762852032): Work completed!
Thread 4 (ID: 125906746066624): Work completed!
Main: Thread 2 has finished
Main: Thread 3 has finished
Main: Thread 4 has finished
Main: Thread 5 has finished
All threads completed!
root@DESKTOP-GFUS3VG:~/Home_tasks_OS_1149/Assignment1-OS#
```

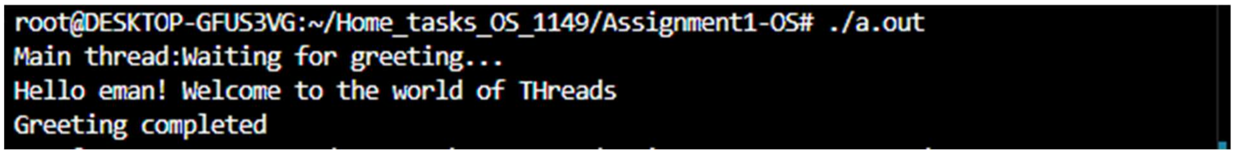
TASK 2:



```
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>

void* thread_func(void* arg){
    char* thread_name=(char*) arg;
    printf("Hello %s! Welcome to the world of THreads \n",thread_name);
    return NULL;
}


int main(){
pthread_t threadd;
char* name="eman";
```

```c
pthread_create(&threadd,NULL,thread_func,name);
printf("Main thread:Waiting for greeting...\n");
pthread_join(threadd,NULL);
printf("Greeting completed\n");
return 0;
}
```

OUTPUT:

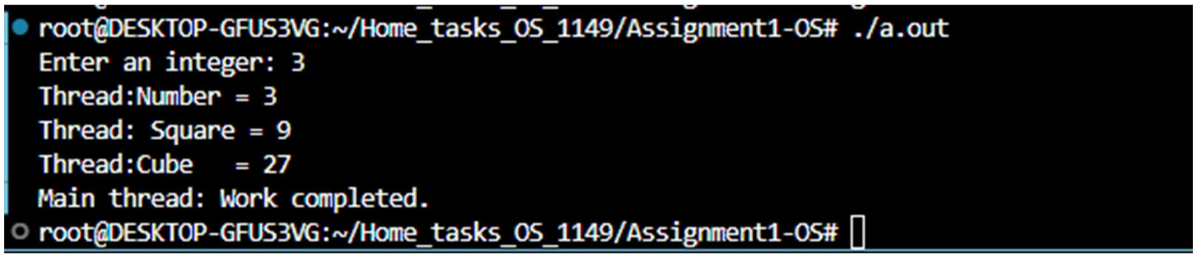


```
root@DESKTOP-GFUS3VG:~/Home_tasks_OS_1149/Assignment1-OS# ./a.out
Main thread:Waiting for greeting...
Hello eman! Welcome to the world of THreads
Greeting completed
```

Task3:

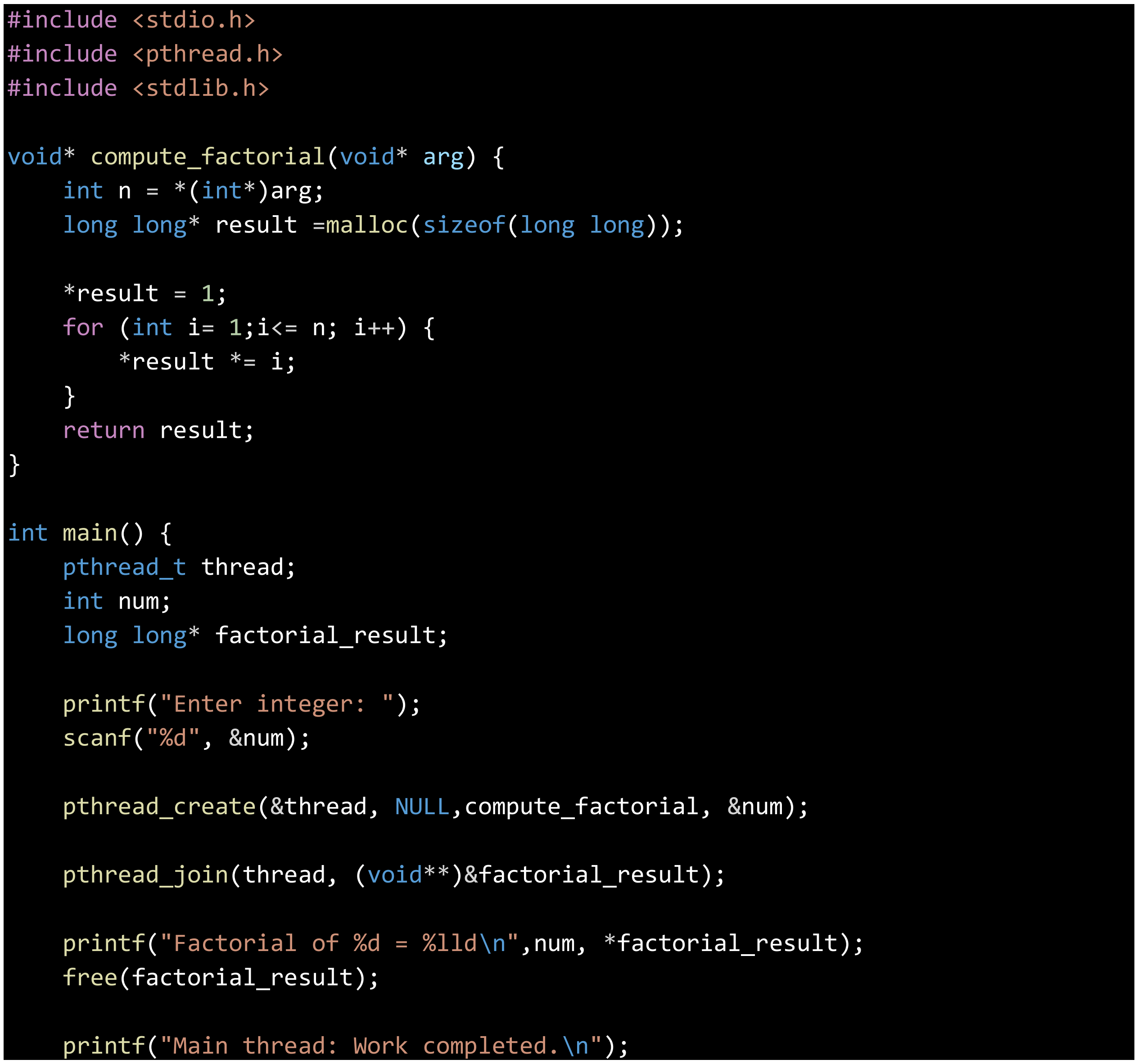
```c
#include <stdio.h>
#include <pthread.h>


void* compute(void* arg) {
    int num = *(int*)arg;
    printf("Thread:Number = %d\n",num);
    printf("Thread: Square = %d\n",num*num);
    printf("Thread:Cube   = %d\n", num*num*num);
    return NULL;
}
int main() {
    pthread_t thread;
    int number;
    printf("Enter an integer: ");
    scanf("%d", &number);
    pthread_create(&thread, NULL, compute, &number);
    pthread_join(thread, NULL);
    printf("Main thread: Work completed.\n");

    return 0;
}
```

OUTPUT:



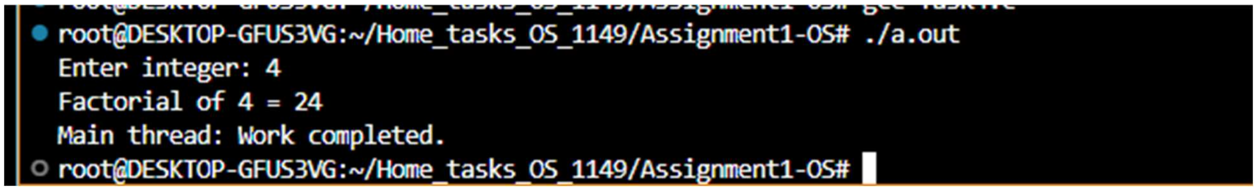
```
root@DESKTOP-GFUS3VG:~/Home_tasks_OS_1149/Assignment1-OS# ./a.out
Enter an integer: 3
Thread:Number = 3
Thread: Square = 9
Thread:Cube   = 27
Main thread: Work completed.
root@DESKTOP-GFUS3VG:~/Home_tasks_OS_1149/Assignment1-OS# 
```

Task 4:



```c
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>

void* compute_factorial(void* arg) {
    int n = *(int*)arg;
    long long* result =malloc(sizeof(long long));

    *result = 1;
    for (int i= 1;i<= n; i++) {
        *result *= i;
    }
    return result;
}

int main() {
    pthread_t thread;
    int num;
    long long* factorial_result;

    printf("Enter integer: ");
    scanf("%d", &num);

    pthread_create(&thread, NULL,compute_factorial, &num);

    pthread_join(thread, (void**)&factorial_result);

    printf("Factorial of %d = %lld\n",num, *factorial_result);
    free(factorial_result);

    printf("Main thread: Work completed.\n");
    return 0;
}
```

OUTPUT:



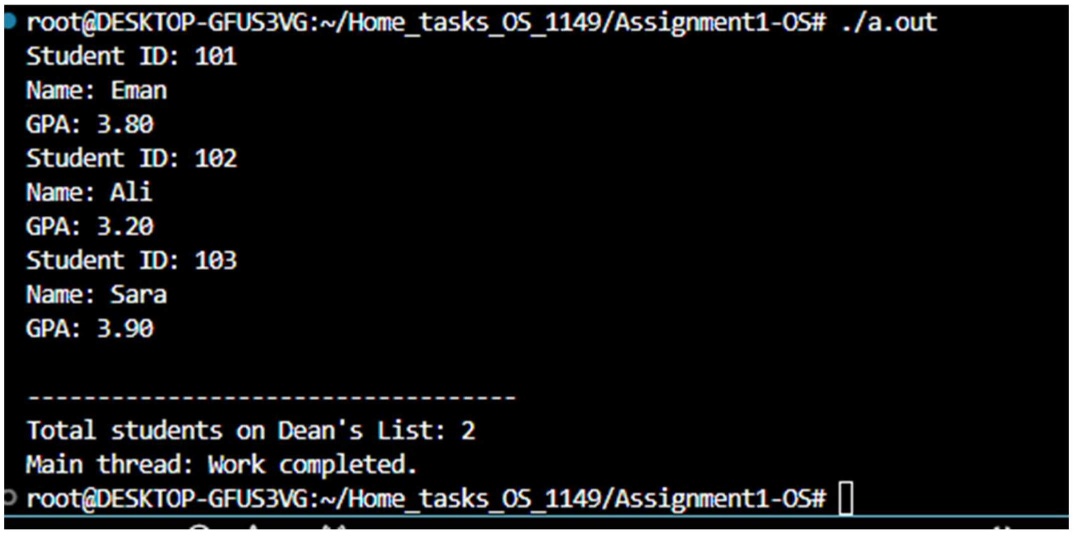
```
root@DESKTOP-GFUS3VG:~/Home_tasks_OS_1149/Assignment1-OS# ./a.out
Enter integer: 4
Factorial of 4 = 24
Main thread: Work completed.
root@DESKTOP-GFUS3VG:~/Home_tasks_OS_1149/Assignment1-OS#
```

Task 5:

```
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <string.h>
typedef struct {
    int student_id;
    char name[50];
    float gpa;
} Student;

void* check_and_print(void* arg) {
    Student* s =(Student*)arg;
    printf("Student ID: %d\n", s->student_id);
    printf("Name: %s\n", s->name);
    printf("GPA: %.2f\n", s->gpa);

    int* eligible = malloc(sizeof(int));
    if (s->gpa >= 3.5) {
        *eligible = 1;
    } else {
        *eligible = 0;
    }
    pthread_exit(eligible);
}

int main() {
    pthread_t threads[3];
    Student students[3] = {
```

```c
        {101, "Eman", 3.8},
        {102, "Ali", 3.2},
        {103, "Sara", 3.9}
    };

    int* eligibility[3];
    int dean_count =0;
    for (int i =0; i < 3; i++) {
        pthread_create(&threads[i], NULL, check_and_print, &students[i]);
    }

    for (int i = 0; i < 3; i++) {
        pthread_join(threads[i], (void**)&eligibility[i]);
        dean_count += *eligibility[i];

    }

    printf("\n-----------------------------------\n");
    printf("Total students on Dean's List: %d\n", dean_count);
    printf("Main thread: Work completed.\n");
    return 0;
}
```

OUTPUT:



```
root@DESKTOP-GFUS3VG:~/Home_tasks_OS_1149/Assignment1-OS# ./a.out
Student ID: 101
Name: Eman
GPA: 3.80
Student ID: 102
Name: Ali
GPA: 3.20
Student ID: 103
Name: Sara
GPA: 3.90

-----------------------------------
Total students on Dean's List: 2
Main thread: Work completed.
root@DESKTOP-GFUS3VG:~/Home_tasks_OS_1149/Assignment1-OS#
```

Section-B: Short Questions

1.Define an Operating System in a single line.

An Operating system is a system software that acts as an interface between hardware and users, managing processes, memory, and system resources.

2. What is the primary function of the CPU scheduler?

The CPU Scheduler selects one process from the ready queue and gives it to CPU for processing .It selects on the bases of scheduling algorithms like FCFS,Round Robin etc

3. List any three states of a process.

1. Ready-waiting to be assigned to CPU.
2. Running: Currently being executed in CPU.
3. Waiting: waiting for an I/O operation or event to complete.

4.What is meant by a Process Control Block?

A PCB is a data structure that stores all important information of a process like its Process ID,CPU registers,memory limits,state and scheduling information.

5. Differentiate between a process and a program.

A program is a set of instructions, while a process is a program in execution.

Program	Process
A passive set of instructions stored on disk.	A process is an active instance of a program that is being executed by the CPU.
Static-it does not perform any action until executed	It represents an ongoing activity.
One program can create multiple processes.	Each process is an independent execution of a program.

6. What do you understand by context switching?

When CPU switches to another process, the system must save the state of the old process and load the saved state for the new process via a context switch. Context of a process represented in the PCB. Context-switch time is pure overhead,the system does no useful work while switching.The more complex the OS and the PCB,the longer the context switch.

7. Define CPU utilization and throughput.

CPU utilization: The percentage of time the CPU is busy/actively executing processes.

Throughput: The total number of processes completed by CPU in a certain time.

8. What is the turnaround time of a process?

Turnaround time is the total time taken from the submission of a process to its completion.It includes the waiting time.

9. How is waiting time calculated in process scheduling?

Waiting time is the total time a process spends in the ready queue waiting for CPU allocation.

Waiting Time=Turnaround Time−Burst Time

10. Define response time in CPU scheduling.

Response time is the time interval between a process's submission and the first time it gets the CPU for execution.

Response Time=Time of first CPU allocation−Arrival Time.

11. What is preemptive scheduling?

It allows the CPU to forcibly take control from a running process if a higher-priority or newly arrived process needs the CPU, like in (SRTF) the process with shortest remaining time replaces the running process with longer time.

Example: Round Robin, Shortest Remaining Time First (SRTF)

12. What is non-preemptive scheduling?

Non-preemptive scheduling means once a process starts executing, it cannot be interrupted until it finishes. Example: First come first serve.

13. State any two advantages of the Round Robin scheduling algorithm.

- Ensures **fairness** — each process gets equal CPU time in turns.
- Provides **better response time** for interactive or time-sharing systems.

14. Mention one major drawback of the Shortest Job First (SJF) algorithm.

The major drawback of SJF is that it can lead to **starvation** (long processes may never get executed if short ones keep arriving).

15. Define CPU idle time.

This is the total time CPU remains unused or waits because there are no ready processes for execution.

16. State two common goals of CPU scheduling algorithms.

- Maximum CPU utilization (throughput)
- Minimum response time, and waiting time for processes.

17. List two possible reasons for process termination.

- The process finishes so it terminated.
- There is an illegal instruction.

18. Explain the purpose of the wait() and exit() system calls.

Exit(): Terminates the current process and gives status code to parent.

Wait(): Allows **a** parent process to wait until one of its child processes finishes, and collects its termination status.

19. Differentiate between shared memory and message-passing models of inter-processcommunication.

Shared memory	Message parsing
Processes share common memory space	Processes communicate by sending and receiving messages.
Processes must **use** synchronization to avoid race conditions.	Synchronization **is** implicit—message exchange ensures that one process waits until the message is received.
Data is exchanged by reading/writing to a common memory segment.	Data is exchanged using **system calls**

20.Differentiate between a thread and a process.

Process	**Thread**
An independent program in execution	A lightweight unit of execution within a process
Has its own address space	Shares address space and resources of the process
High overhe,needs full context switch	Low overheadad
One process crach does not affect other process	One thread crash does not effect other thread but can terminate the whole program

22. Define multithreading.

Multithreading is the capability of a process to execute multiple threads concurrently within the same address space, allowing parallelism and better CPU utilization.

23. Explain the difference between a CPU-bound process and an I/O-bound process.

CPU-bound	**I/O-bound**
Spends most time performing computations.	Spends most time waiting for I/O operations.
High CPU utilization.	Low CPU utilization.
Few.	Frequent.

24. What are the main responsibilities of the dispatcher?

The **dispatcher** is a module of the OS responsible for:

1. Context switching between processes. It's responsible for transferring CPU control quickly between processes.
2. Switching CPU from one process to another.It ensures smooth and efficient transitions so the CPU is never idle unnecessarily.
3. Mode switching between user mode and kernel mode.

25. Define starvation and aging in process scheduling.

Starvation:A situation when a process waits in ready queue because other higher priority tasks are being executed and they keep getting CPU time.

Aging:A technique to prevent starvation.This is done by increasing a process's priority as it waits longer in the queue.

26. What is a time quantum (or time slice)?

A **time quantum** is the **fixed amount of CPU time** allocated to each process in **Round Robin scheduling** before it is preempted and moved back to the ready queue.Each process can utiliza CPU for time quantum.

27. What happens when the time quantum is too large or too small?

Too Large: It reduces context switching.mostly process terminate when finished.

Too Small: Increases context switching overhead and reduces CPU efficiency.

28. Define the turnaround ratio (TR/TS).

The **Turnaround Ratio** is the ratio of turnaround time (TR) to service time (TS):

Turnaround ratio=Turnaround Time/Service Time

It indicate how much longer did process took compared to actual CPU execution.

29. What is the purpose of a ready queue?

The ready queue holds all processes that are in main memory and ready to execute but currently waiting for CPU allocation

30. Differentiate between a CPU burst and an I/O burst.

CPU burst: Period of time when process is executing instructions on CPU.

I/O burst: Period when process waits for input/output operations to complete.

31. Which scheduling algorithm is starvation-free, and why?

Its FCFS.This is because every process is served in order of arrival .no process is skipped or ignored. Once a process enters the ready queue, it will eventually get CPU time.

32. Outline the main steps involved in process creation in UNIX.

fork () – Creates a child process (duplicate of parent).

exec () – Replaces the child's memory with a new program.

wait () – Parent waits for the child to finish.

exit() – Child terminates and returns status to the parent

33. Define zombie and orphan processes.

Zombie process:A process that has finished execution but still has an entry in the table because the parent hasn't called wait().

Orphan process:A process whose parent has terminated before it .it's later adopted by the init process.

34. Differentiate between Priority Scheduling and Shortest Job First (SJF).

Priority Scheduling	Shortest Job First (SJF).
Process is selected Based on process priority value.	Process selected based on shortest CPU burst time.
Preemptive	Can be preemptive or non-preemptive.
Low priority tasks can suffer starvation	Long processes can suffer starvation.

35. Define context switch time and explain why it is considered overhead.

Context switch time is the time required by the OS to save the state of the current process and load the state of the next process.It's considered overhead because it consumes CPU time but performs no useful computation for user processes.

36. List and briefly describe the three levels of schedulers in an Operating System.

Long term scheduler: The long-term scheduler determines which programs are admitted to the system for processing. Thus, it controls the degree of multiprogramming.

Short Term Scheduler: Selects which ready process will execute next on the CPU.

Medium Term Scheduler: Manages process swapping between main memory and secondary storage (suspension/resumption).

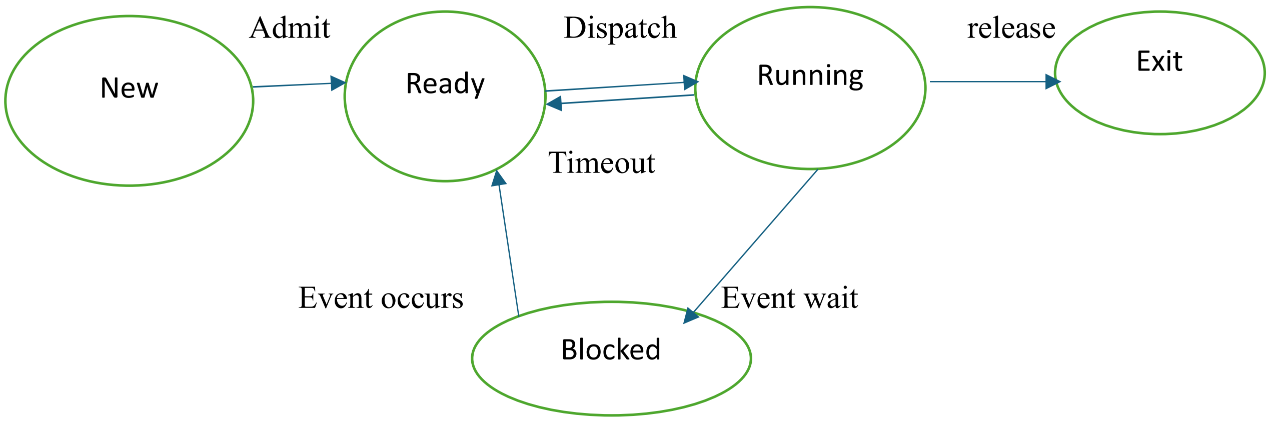
37. Differentiate between User Mode and Kernel Mode in an Operating System.

User Mode	Kernel Mode
Mode in which user applicatins run with limited access to system resources.	Mode in which the operating system kernel executes with full access to all hardware and memory.
Access level Restricted — cannot directly access hardware or memory.	Can **access** all system resources, including hardware and kernel memory.

Must use **system calls** to request OS services.	Executes **directly** system call handlers to perform the requested operations.
Errors affect only the process, not the whole system.	Error can crash the whole system.

Section-C: Technical / Analytical Questions (4 marks each)

1. Describe the complete life cycle of a process with a neat diagram showing transitions.between New, Ready, Running, Waiting, and Terminated states.



New: The process is created by OS. A new memory is allocated and resources are assigned.

Ready: The process is loaded into main memory and is ready to execute, but CPU is busy with another process.

Running: The process is currently executing instructions on the CPU.

Running -→ waiting: if it needs I/O or waits for an event.

Running → Ready: if preempted by scheduler due to time slice or higher priority

Waiting: The process cannot continue until a specific event occurs.

Terminated: The process has finished execution or has been killed.

2. Write a short note on context switch overhead and describe what information must be saved and restored.

A context switch occurs when the CPU switches from executing one process (or thread) to another.**Context switch overhead** is the time and CPU work spent by the operating system during the switching process.During this time, no useful work is done the CPU is busy saving and restoring data structures,no actual computation for user program is done. this is why it is considered system overhead.

It involves:

- **Saving** the state (context) of the currently running process.
- **Loading** the saved state of the next process to be executed.

Information saved in PCB:

- CPU registers.
- Program counter.
- Stack pointer.
- I/O status information.
- Process state.
- Scheduling information

3. List and explain the components of a Process Control Block (PCB).

A Process Control Block (PCB) is a data structure maintained by the operating system that contains all essential information about a specific process.

Components of PCB:

1.Process ID: Unique identifier assigned by the OS to each process.

2.Process state: Indicates the current status of the process — e.g., New*,* Ready*,* Running*,* Waiting*, or* Terminated*.*

3.CPU Registers: Includes all CPU register contents.

4.Program Counter: Holds the address of the next instruction to execute. When the process is resumed, the OS uses this value to continue from where it left off.

5.CPU scheduling information: Includes process priority**,** scheduling queue pointers**.**

6. I/O Status Information: Contains list of open files, I/O devices assigned.

- When process is **created**, OS allocates PCB and fills in initial info.
- When process is **executing**,OS updates PCB continuously.
- During **context switch** ,PCB saves the current process state; OS loads another process's PCB.
- When process **terminates** , PCB is removed and resources are released.

4. Differentiate between Long-Term, Medium-Term, and Short-Term Schedulers with examples.

Schedulers are OS components that select processes at different stages of execution.

Long term scheduler	**Medium Term Scheduler**	**Short Term Scheduler**
Decides which new processes are loaded from secondary storage (disk) into main memory.	Handles swapping — temporarily removes (suspends) processes from main memory and moves them to secondary storage (swap space).	Decides which ready process will run next on the CPU
Example: In a batch processing system, the OS decides which jobs from the job pool (on disk) to admit into memory.	**Example:** In Linux, when memory is low, the kernel may swap out a suspended background process to disk.	**Example:** Algorithms like Round Robin, SJF are implemented by this scheduler.

5. Explain CPU Scheduling Criteria (Utilization, Throughput, Turnaround, Waiting, and Response) and their optimization goals.

CPU Scheduling Criteria are the performance metrics used by the operating system to evaluate how good or efficient a CPU scheduling algorithm is.

Utilization: Measures how effectively the CPU is being used. It is the percentage of time the CPU is busy executing processes.This should be **maximized**.

Throughput: Number of processes completed per unit time**.** This should be **maximized**.

Turnaround time: Total time taken by a process from submission to completion. Includes waiting, execution, and I/O times. This should be **minimized.**

Waiting time: Total time a process spends waiting in the ready queue before getting CPU time. This should be **minimized.**

Response Time: Time between process submission and the first response (first time it gets CPU).This should be **minimized.**

Section-D: CPU Scheduling Calculations

• Perform the following calculations for each part (A–C).

a) Draw Gantt charts for FCFS, RR (Q=4), SJF, and SRTF.

b) Compute Waiting Time, Turnaround Time, TR/TS ratio, and CPU Idle Time.

c) Compare average values and identify which algorithm performs best.

Part-A

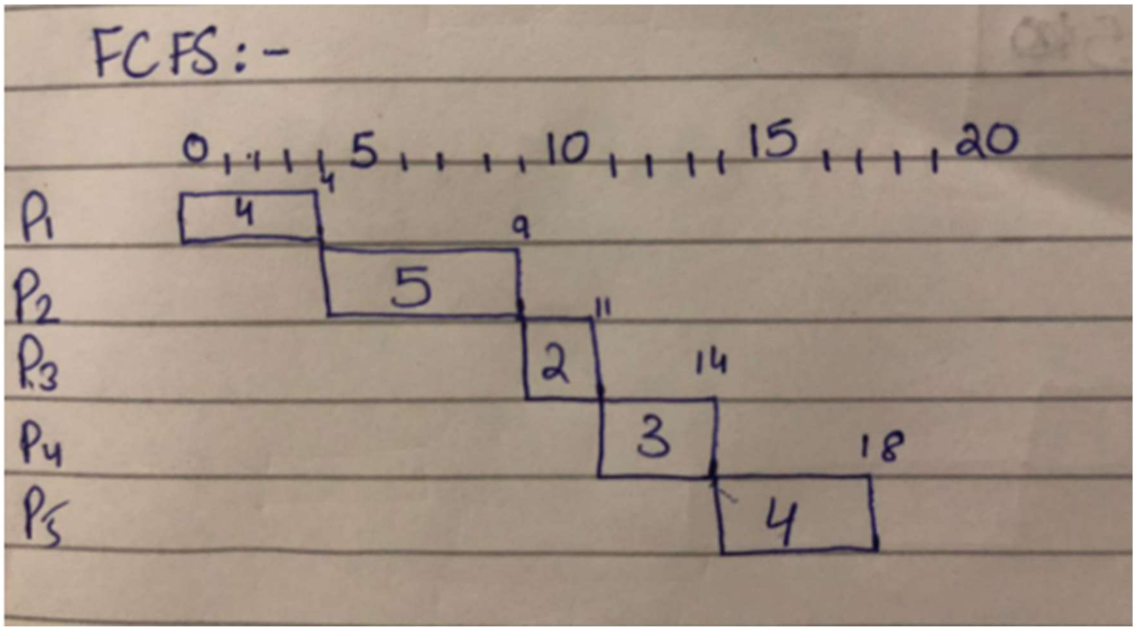
Process	Arrival Time	Service Time
P1	0	4
P2	2	5
P3	4	2
P4	6	3
P5	9	4

Part-A

Process	Arrival Time	Service Time
P1	0	4
P2	2	5
P3	4	2
P4	6	3
P5	9	4

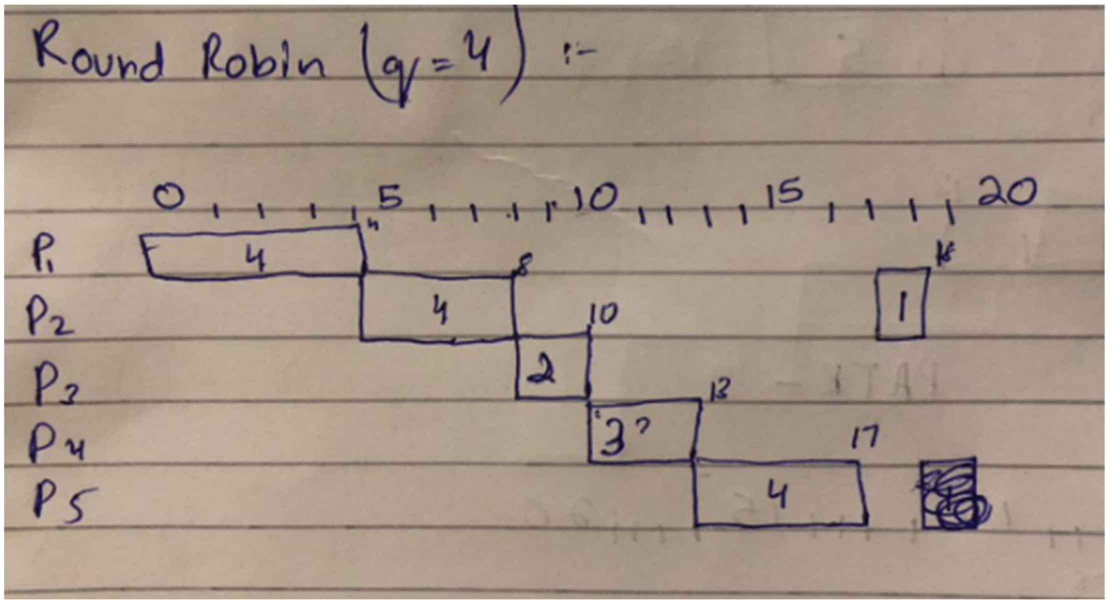
FCFS

	P1	P2	P3	P4	P5
Waiting time	0	2	5	5	5
Turnaround Time	4	7	7	8	9
TR/TS ratio	1	1.4	3.5	2.67	2.25



Round robbin:

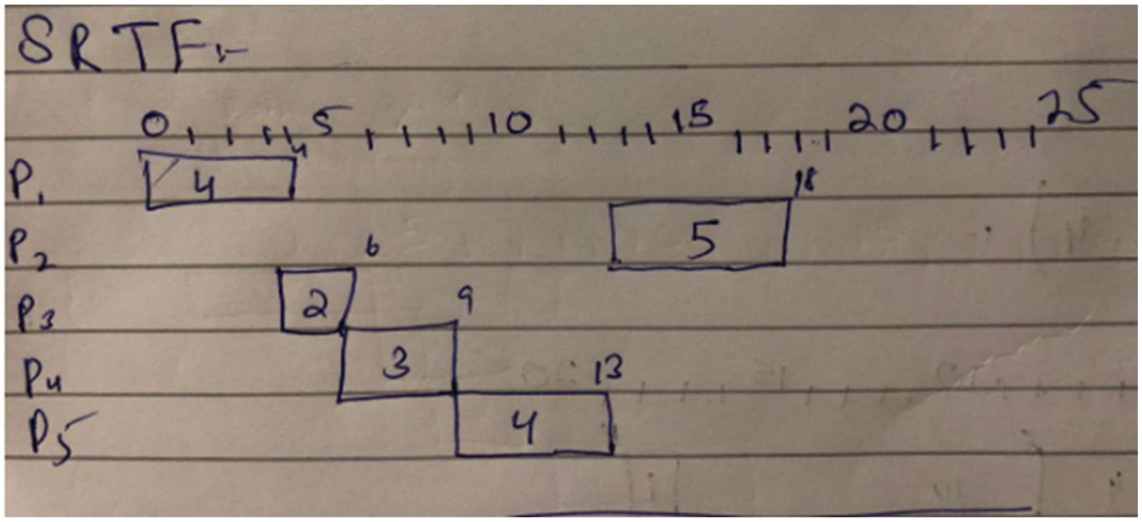
	P1	P2	P3	P4	P5
Waiting time	0	2	4	4	4
Turnaround Time	4	16	6	7	8
TR/TS ratio	1	3.2	3	2.33	2



Shortest job first:

	P1	P2	P3	P4	P5
Waiting time	0	11	0	0	0
Turnaround Time	4	16	2	3	4
TR/TS ratio	1	3.2	1	1	1

Shortest remaining job first:



	P1	P2	P3	P4	P5
Waiting time	0	11	0	0	0
Turnaround Time	4	16	2	3	4
TR/TS ratio	1	3.2	1	1	1

c) Compare average values and identify which algorithm performs best.

FCFS:7

Round robin:8.2

Shortest job first:5.8

Shortest remaining job first:5.8

Both SPN and SRJF perform well.

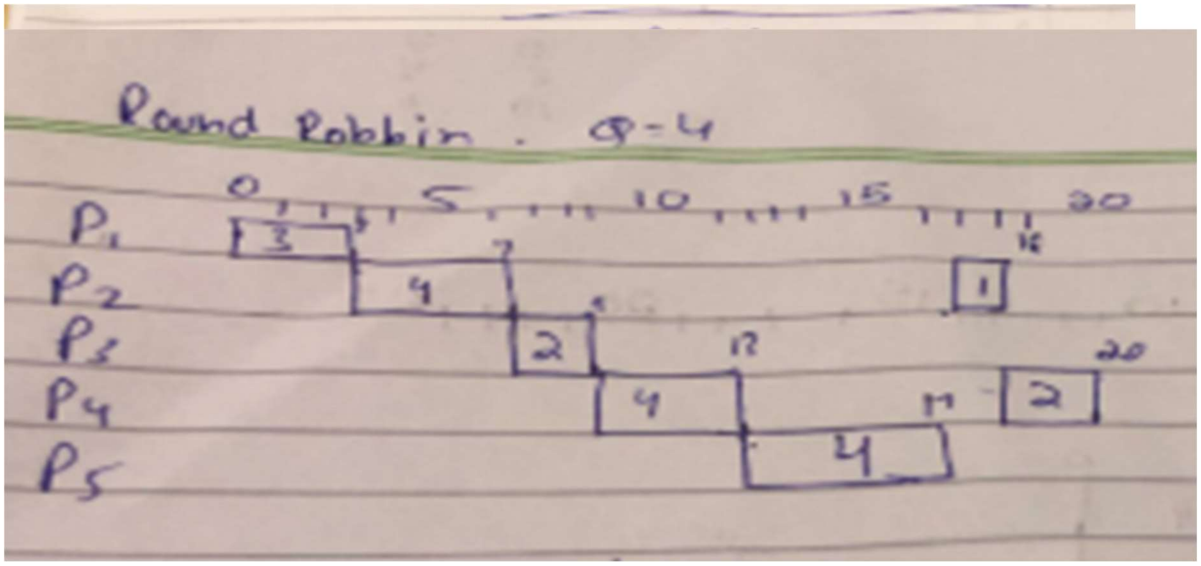
Part B:

Part-B

Process	Arrival Time	Service Time
P1	0	3
P2	1	5
P3	3	2
P4	9	6
P5	10	4

First Come First Serve:

	P1	P2	P3	P4	P5
Waiting time	0	2	5	1	6
Turnaround Time	3	7	7	7	10
TR/TS ratio	1	1.4	3.5	1.16	2.5

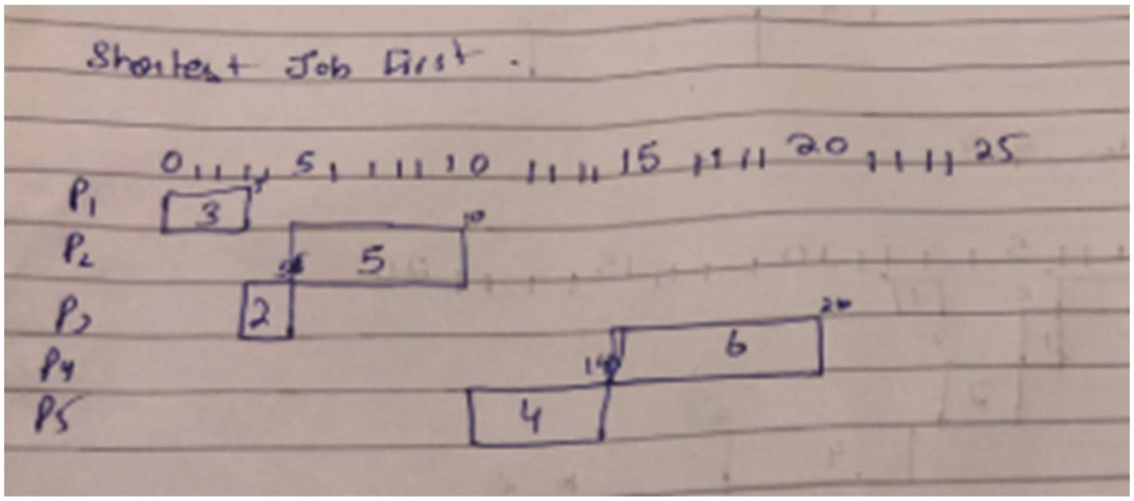


Round Robbin:

	P1	P2	P3	P4	P5
Waiting time	0	2	4	0	3
Turnaround Time	3	17	6	11	7
TR/TS ratio	1	3.4	3	1.83	1.75

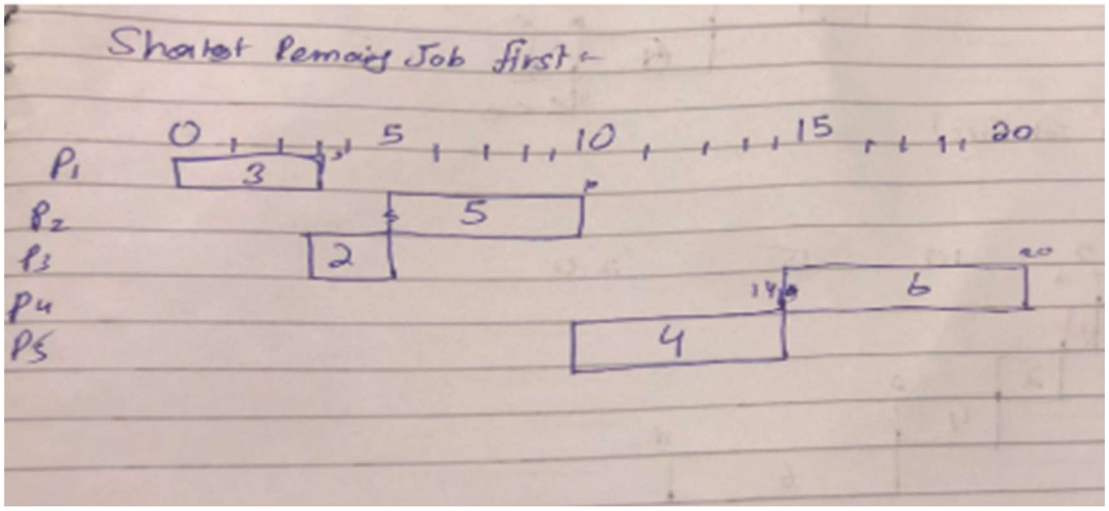
Shortest Job first:

	P1	P2	P3	P4	P5
Waiting time	0	4	0	5	0
Turnaround Time	3	9	2	14	10
TR/TS ratio	1	1.8	1	2.33	2.5



Shortest remaining job first:

	P1	P2	P3	P4	P5
Waiting time	0	4	2	5	0
Turnaround Time	3	9	2	11	4
TR/TS ratio	1	1.8	1	1.83	1



c) Compare average values and identify which algorithm performs best.

FCFS:6.8

Round robin:8.8

Shortest job first:7.6

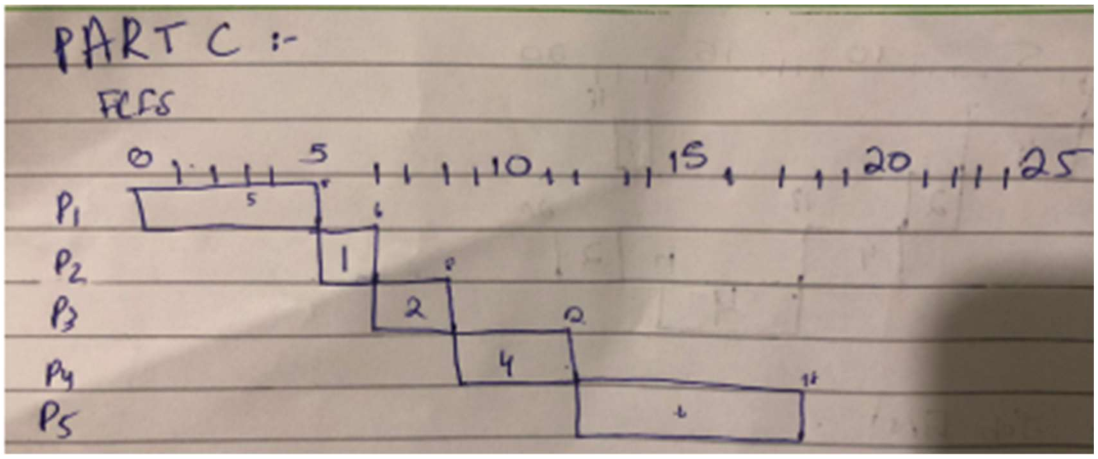
Shortest remaining job first:5.8

SRJF perform well

PART-C:

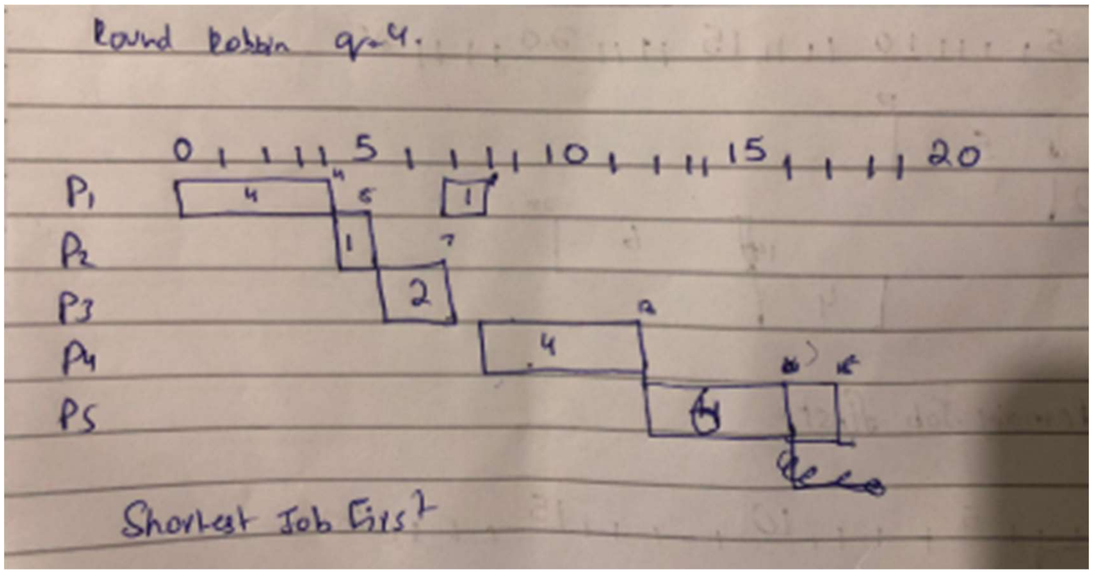
Process	Arrival time	Service time
P1	0	5
P2	3	1
P3	5	2
P4	8	4
P5	10	6

FCFS:



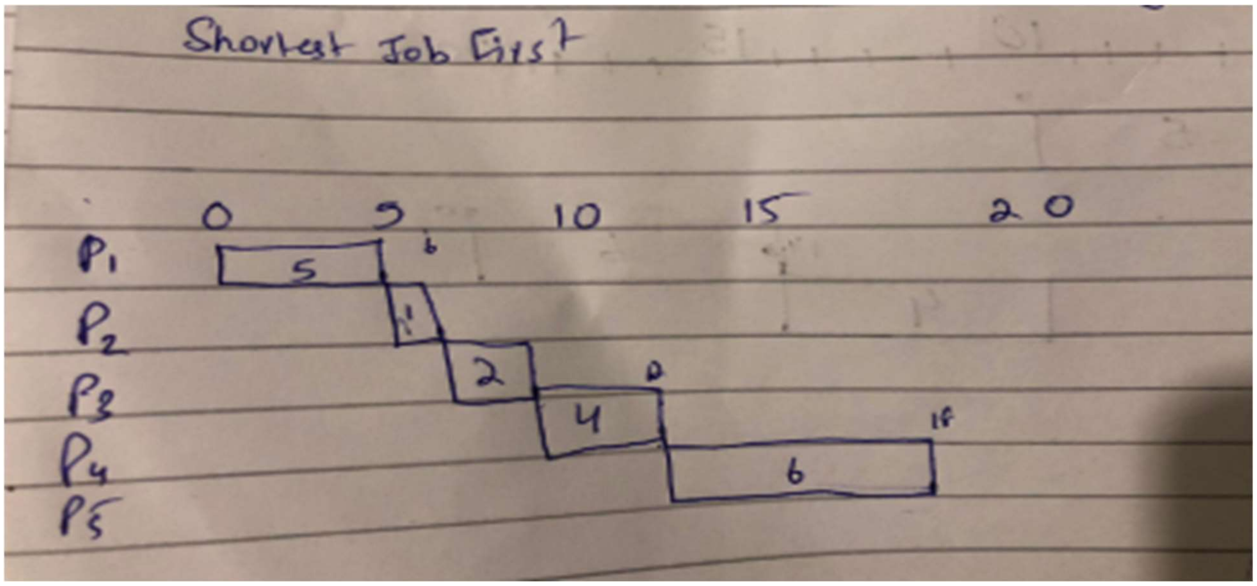
	P1	P2	P3	P4	P5
Waiting time	0	2	1	2	4
Turnaround Time	5	4	4	6	10
TR/TS ratio	1	4	2	1.5	2.5

Round Robbin:



	P1	P2	P3	P4	P5
Waiting time	0	1	0	0	2
Turnaround Time	8	2	5	4	8
TR/TS ratio	1.6	2	2.5	1	1.33

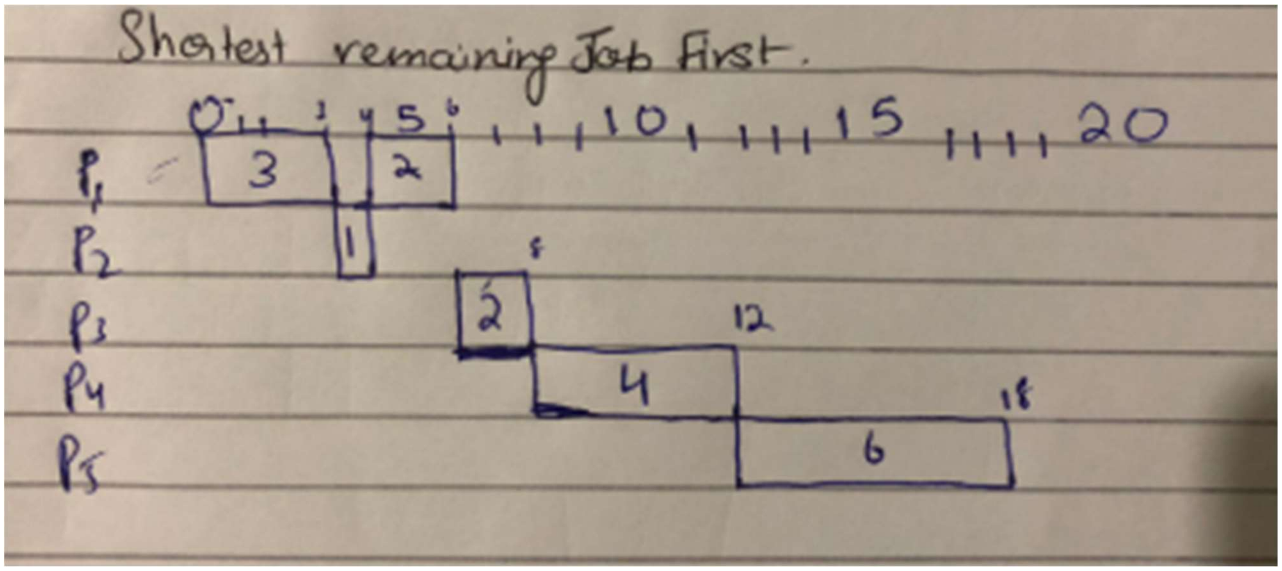
Shortest Job First:



	P1	P2	P3	P4	P5

Waiting time	0	2	1	0	2
Turnaround Time	5	3	3	4	8
TR/TS ratio	1	3	1.5	1	1.33

Shortest Remaining job first:



	P1	P2	P3	P4	P5
Waiting time	0	0	1	0	2
Turnaround Time	6	1	2	4	8
TR/TS ratio	1.2	1	1	1	1.33

c) Compare average values and identify which algorithm performs best.

FCFS:7

Round robin:5.8

Shortest job first:4.6

Shortest remaining job first:4.2

SRJF performs well.