National Textile University, Faisalabad

Department of Computer Science
Course: Operating Systems (CSC-3075)
Semester: Fall 2025
BSSE 5th A
Instructor: Mr. Nasir Mahmood
Submitted By: Shanza Batool
Reg. No:23-NTU-CS-1209
Submission Date:26th October,2025

Assignment-1

Section-A: Programming Tasks

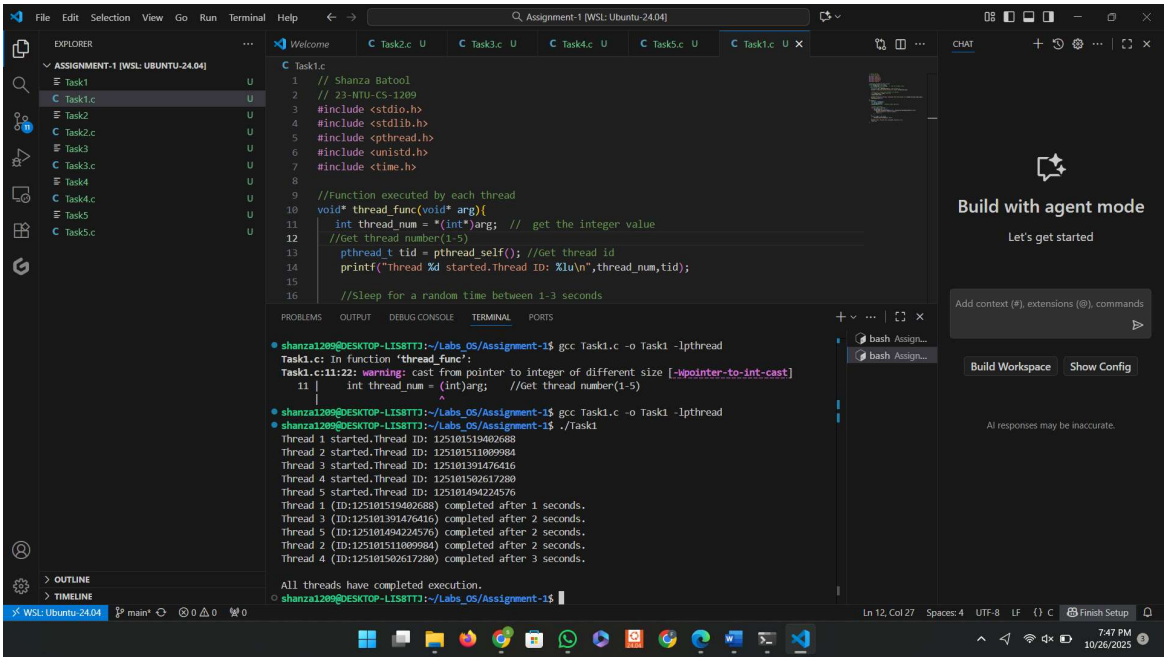
- Instructions:

- Complete all tasks in C using the pthread library.
- Properly comment on your code and include your name, registration number, and task title at the top of each file.
- Use clear screenshots of both code and execution output (CodeSnap preferred).

Task 1 – Thread Information Display

Write a program that creates 5 threads. Each thread should:
• Print its thread ID using `pthread_self()`.
• Display its thread number (1st, 2nd, etc.).
• Sleep for a random time between 1–3 seconds.
• Print a completion message before exiting.

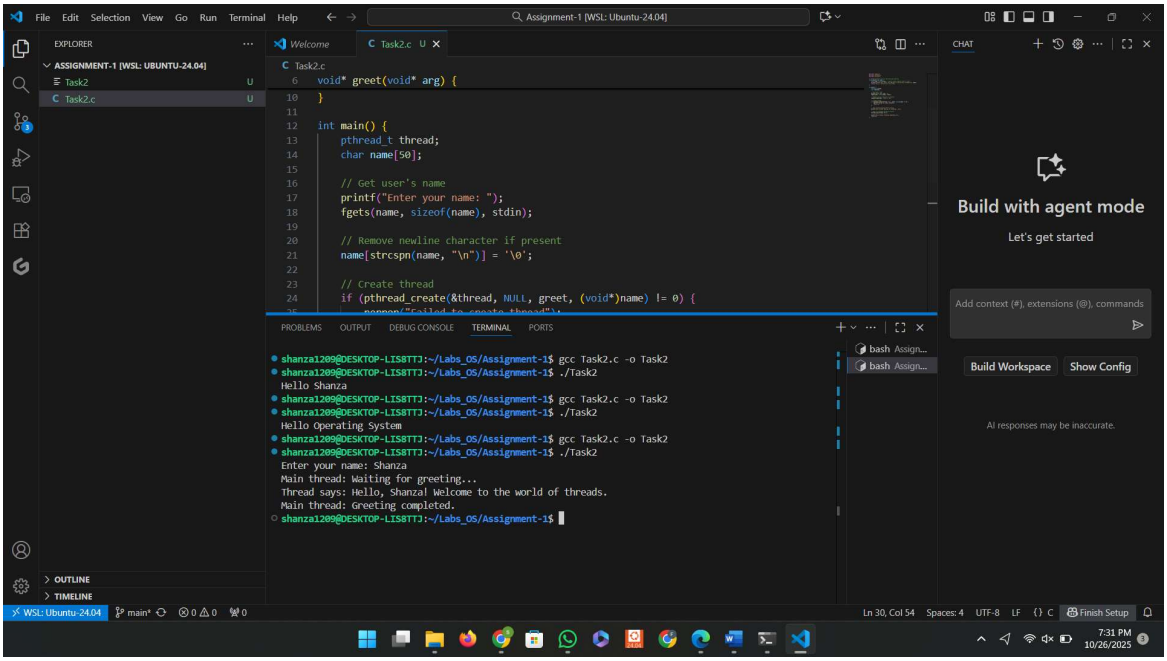
Expected Output: Threads complete in different orders due to random sleep times.



Task 2 – Personalized Greeting Thread

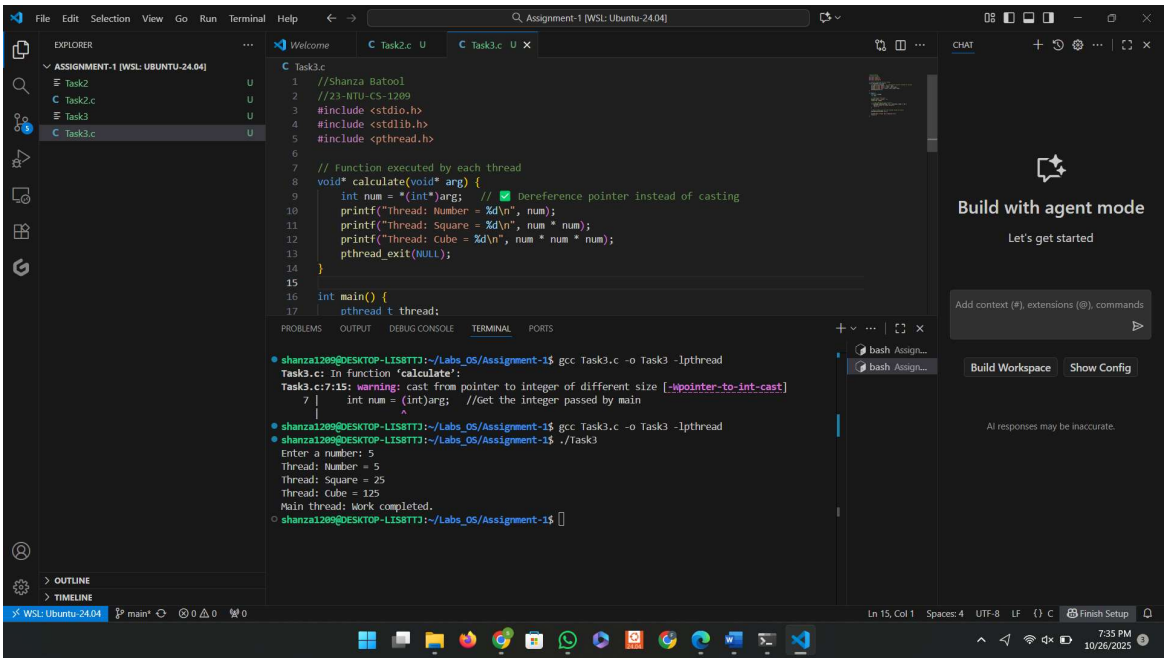
Write a C program that:
- Creates a thread that prints a personalized greeting message.
- The message includes the user's name passed as an argument to the thread.
- The main thread prints "Main thread: Waiting for greeting..." before joining the created thread.

Example Output:
Main thread: Waiting for greeting...
Thread says: Hello, Ali! Welcome to the world of threads. Main thread: Greeting completed.



Task 3 – Number Info Thread

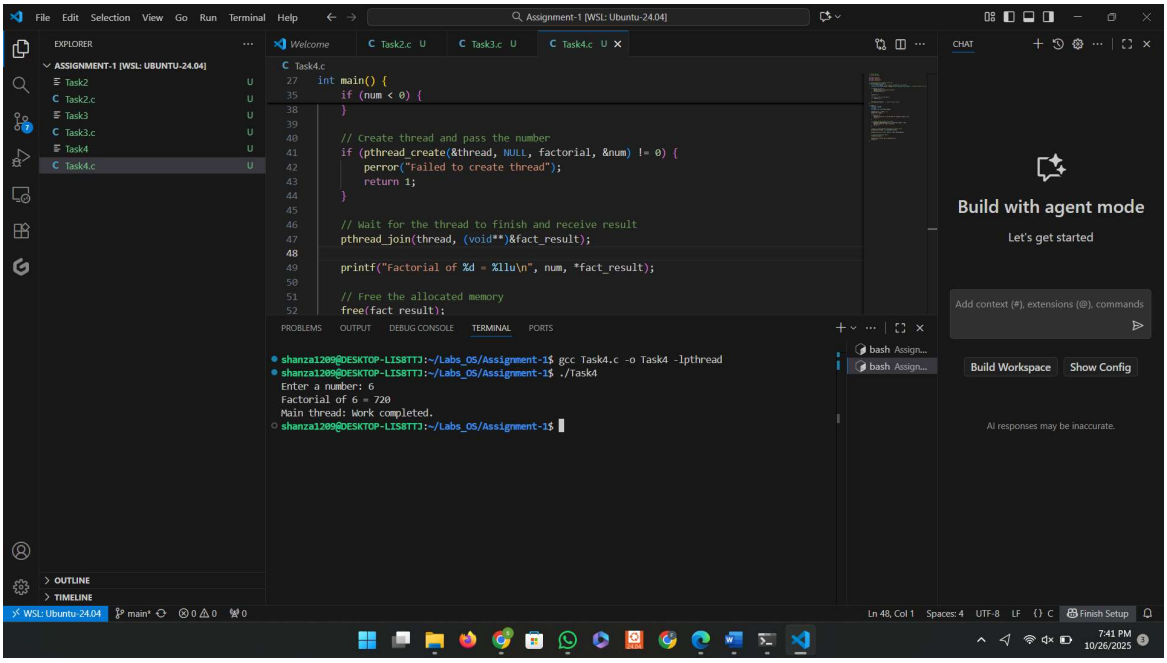
Write a program that:
- Takes an integer input from the user.
- Creates a thread and passes this integer to it.
- The thread prints the number, its square, and cube.
- The main thread waits until completion and prints "Main thread: Work completed."



Task 4 – Thread Return Values

Write a program that creates a thread to compute the factorial of a number entered by the user.

- The thread should return the result using a pointer.
- The main thread prints the result after joining.

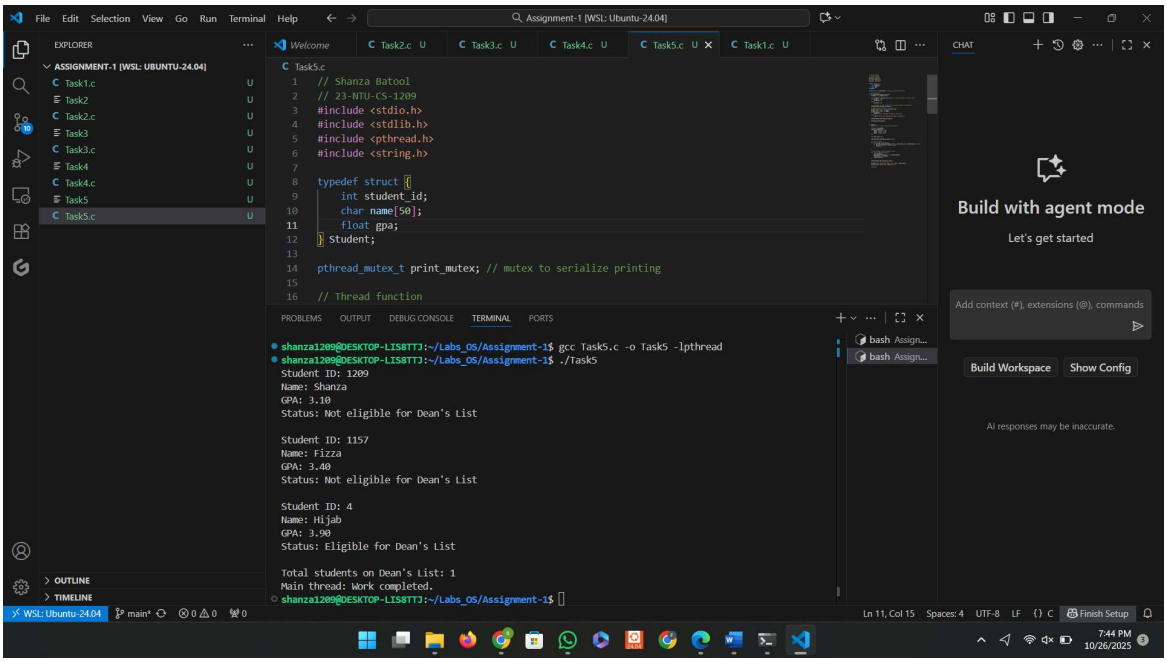


Task 5 – Struct-Based Thread Communication

Create a program that simulates a simple student database system.

- Define a struct: `typedef struct { int student_id; char name[50]; float gpa; } Student;`
- Create 3 threads, each receiving a different Student struct.
- Each thread prints student info and checks Dean's List eligibility (GPA ≥ 3.5).

• The main thread counts how many students made the Dean's List.



Section-B: Short Questions

1. Answer all questions briefly and clearly.
2. **Define an Operating System in a single line.**
   Operating system is a system software which acts as intermediary between user and hardware and Provides environment for program execution.

3. **What is the primary function of the CPU scheduler?**
   The **CPU scheduler** selects which process will run next on the CPU.

4. **List any three states of a process.**
   - Ready
   - Running
   - Waiting(Blocked)

5. **What is meant by a Process Control Block (PCB)?**
   A **Process Control Block (PCB)** is a data structure that stores all information about a process, such as its state, registers, and ID.

6. **Differentiate between a process and a program.**
   A **program** is a passive set of instructions; a **process** is an active instance of a program in execution.

7. **What do you understand by context switching?**
   Context switching is the process of saving the current state of a process and loading the state of another process.

8. **Define CPU utilization and throughput.**
   - **CPU utilization:** Percentage of time the CPU is busy.
   - **Throughput:** Number of processes completed per unit time.

9. **What is the turnaround time of a process?**
   Turnaround time is the total time taken from process submission to its completion.

10. **How is waiting time calculated in process scheduling?**
    Waiting time = Turnaround time – Burst(Service) time.

11. **Define response time in CPU scheduling.**
    Response time is the time from process submission until the first response is produced.

12. **What is preemptive scheduling?**
    Preemptive scheduling allows the CPU to be taken from a process before it finishes.

13. **What is non-preemptive scheduling?**
    Non-preemptive scheduling allows a process to run until it completes or blocks voluntarily.

14. **State any two advantages of the Round Robin scheduling algorithm.**
    - Fair CPU allocation among processes.
    - Good response time for interactive systems.

15. **Mention one major drawback of the Shortest Job First (SJF) algorithm.**
    It can cause **starvation** for long processes.

16. **Define CPU idle time.**
    CPU idle time is the time when the CPU is not executing any process.

17. **State two common goals of CPU scheduling algorithms.**
    - Maximize CPU utilization.
    - Minimize waiting and turnaround times.

18. **List two possible reasons for process termination.**
    - Normal completion.
    - Error or resource failure.

19. **Explain the purpose of the `wait()` and `exit()` system calls.**
    - **wait():** Makes a parent process wait for child process completion.
    - **exit():** Terminates a process and returns status to the parent.

20. **Differentiate between shared memory and message-passing models of inter-process communication.**
    - **Shared memory:** Processes communicate by reading/writing common memory space.
    - **Message passing:** Processes communicate by sending and receiving messages.

21. **Differentiate between a thread and a process.**
    - **Thread:** A lightweight subprocess sharing memory.
    - **Process:** An independent program with its own memory space.

22. **Define multithreading.**
    Multithreading allows multiple threads to run concurrently within a single process.

23. **Explain the difference between a CPU-bound process and an I/O-bound process.**
    - **CPU-bound process:** Spends more time on computation.
    - **I/O-bound process:** Spends more time on input/output operations.

24. **What are the main responsibilities of the dispatcher?**
    The **dispatcher** loads the selected process onto the CPU and performs context switching.

25. **Define starvation and aging in process scheduling.**
    - **Starvation:** A process never gets CPU time.
    - **Aging:** Gradual increase in process priority to avoid starvation.

26. **What is a time quantum (or time slice)?**
    A **time quantum (time slice)** is the fixed CPU time given to each process in Round Robin scheduling.

27. **What happens when the time quantum is too large or too small?**
    - Too **large:** Poor response time.
    - Too **small:** High context switch overhead.

28. **Define the turnaround ratio (TR/TS).**
    It is the ratio of the total time taken to execute a process (**Turnaround Time**) to the actual CPU time required by that process (**Service Time**). It indicates how efficiently a process is executed; a lower ratio means better performance.

    Turnaround ratio (TR/TS) = Turnaround time / Service (burst) time.

29. **What is the purpose of a ready queue?**
    The **ready queue** holds all processes waiting to be assigned to the CPU.

30. **Differentiate between a CPU burst and an I/O burst.**
    - **CPU burst:** Period when a process uses CPU.
    - **I/O burst:** Period when a process waits for I/O operations.

31. **Which scheduling algorithm is starvation-free, and why?**
    Round Robin is starvation-free because each process gets an equal time slice.

32. **Outline the main steps involved in process creation in UNIX.**
    Steps in **UNIX process creation:**
    - Fork() creates a child process.
    - Child uses exec() to load a new program.
    - Parent waits using wait().

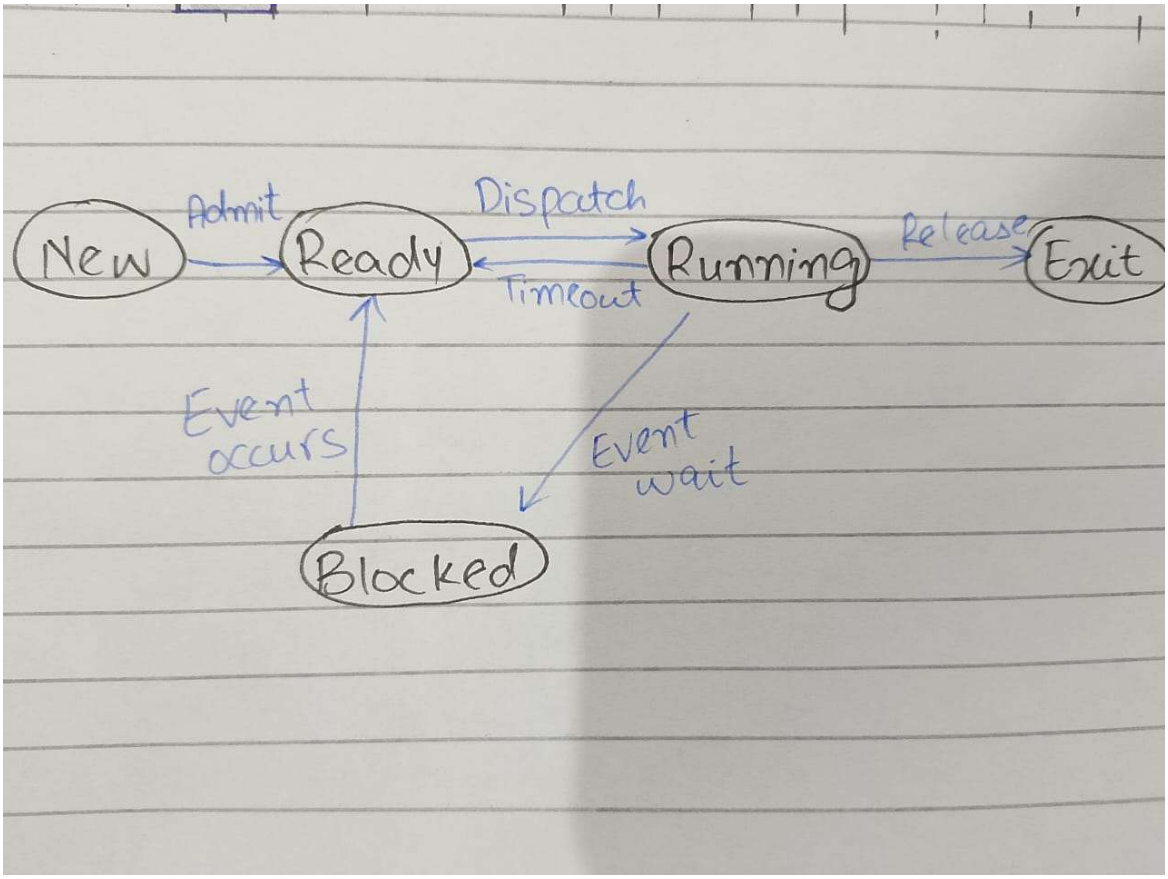
33. **Define zombie and orphan processes.**
    - **Zombie process:** Finished but parent hasn't read its exit status.
    - **Orphan process:** Parent has terminated before the child.

34. **Differentiate between Priority Scheduling and Shortest Job First (SJF).**
    - **Priority Scheduling:** Based on process priority.
    - **SJF:** Based on shortest CPU burst time.
35. **Define context switch time and explain why it is considered overhead.**
    Context switch time is the time taken to switch between processes; it's overhead because no useful work

36. **List and briefly describe the three levels of schedulers in an Operating System.**
    Three **levels of schedulers:**
    1. **Long-term scheduler:** Selects which processes to admit.
    2. **Short-term scheduler:** Chooses which process to run next.
    3. **Medium-term scheduler:** Suspends and resumes processes.
37. **Differentiate between User Mode and Kernel Mode in an Operating System**.
    - **User Mode:** Limited access, runs user applications.
    - **Kernel Mode:** Full access, runs OS code and manages hardware.

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.



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 changes from executing one process to another. During this switch, the CPU must save the state (context) of the current process and load

the state of the next process.
This switching time is called **context switch overhead**, as no useful computation is done during it.

- Time dependent on hardware support
- Some hardware provides multiple sets of registers per CPU ➔ multiple contexts loaded at once.

Information saved and restored includes:

- CPU registers
- Program counter (next instruction address)
- Process state
- Stack pointer
- Memory management information (page tables, base/limit registers)
- Accounting and scheduling information

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

- Process state – running, waiting, etc.
- Program counter – location of instruction to next execute.
- CPU registers – contents of all process centric registers.
- CPU scheduling information- priorities, scheduling queue pointers.
- Memory-management information – memory allocated to the process
- Accounting information – CPU used, clock time elapsed since start, time limits.
- I/O status information – I/O devices allocated to process, list of open file

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

Long-Term Scheduler (Job Scheduler):

- Selects processes from secondary storage (job pool) and loads them into memory
- Example: In batch systems, decides which jobs to load from job pool.
- Executes less frequently
- Slowest of all schedulers.
- Controls the degree of multiprogramming

Medium-Term Scheduler(Swapper):

- Explain CPU Scheduling Criteria (Utilization, Throughput, Turnaround, Waiting, and Response) and their optimization goals.
- Example: Swapping out a process when memory is low.
- Executes occasionally
- Intermediate speed between long and short term schedulers
- Reduces the degree of multiprogramming when

Short-Term Scheduler (Job Scheduler):

- Selects from ready processes the one to execute next on the CPU.
- Example: Round Robin or Priority scheduling in multitasking OS.
- Executes very frequently
- Fastest; runs most frequently.
- Gives less control over the degree of multiprogramming

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

CPU Scheduling Criteria:

- **CPU utilization –** keep the CPU as busy as possible.
- **Throughput –** Number of processes that complete their execution per time unit .
- **Turnaround time –** amount of time to execute a particular process .
- **Waiting time –** amount of time a process has been waiting in the ready queue
- **Response time –** amount of time it takes from when a request was submitted until the first response is produced.

Optimization Goals:

- Max CPU utilization
- Max throughput
- Min turnaround time
- Min waiting time
- Min response time

Section-D: CPU Scheduling Calculations

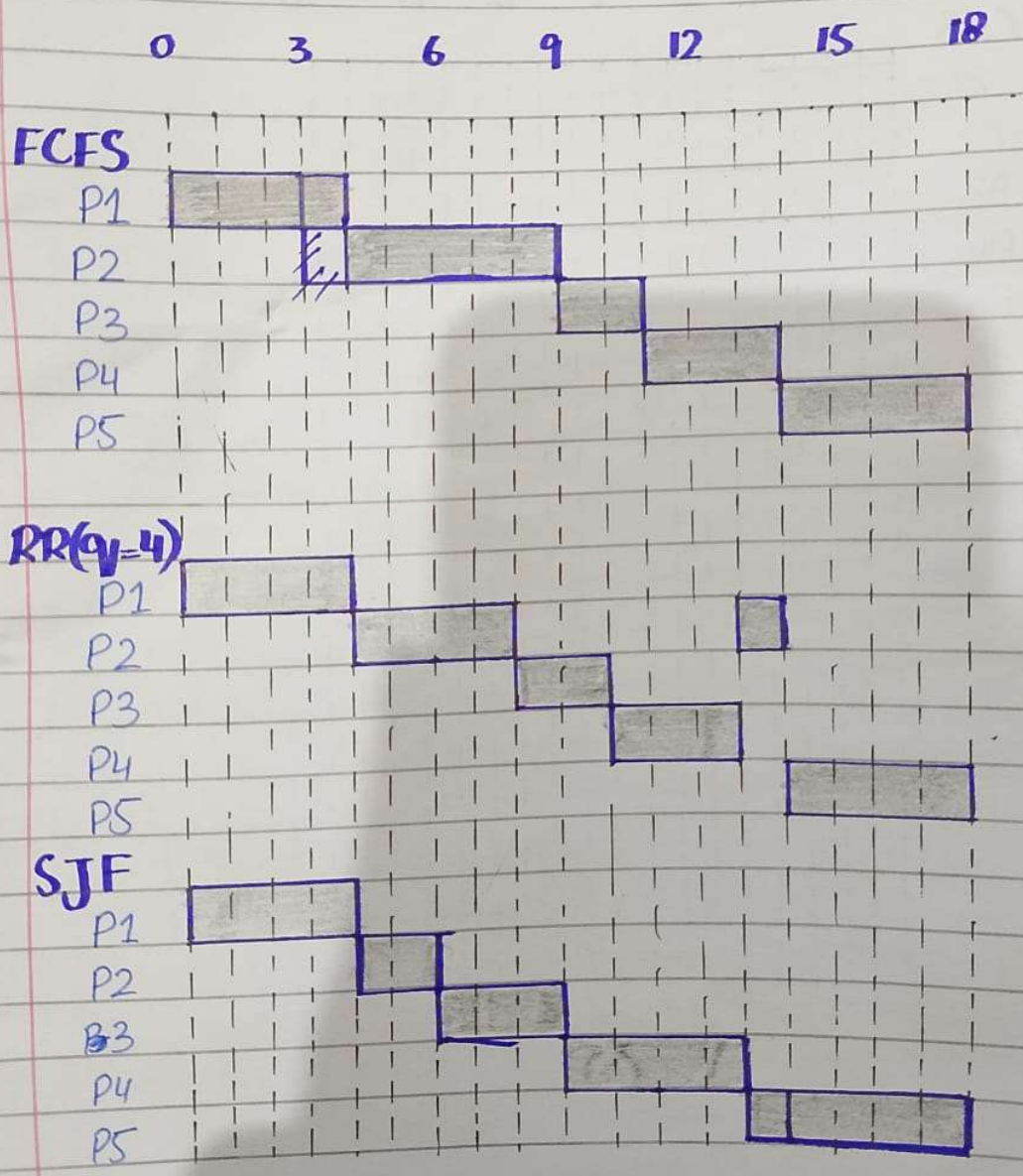
- Perform the following calculations for each part (A–C).
- a) Draw Gantt charts for FCFS, RR (Q=4), SJF, and SRTF.

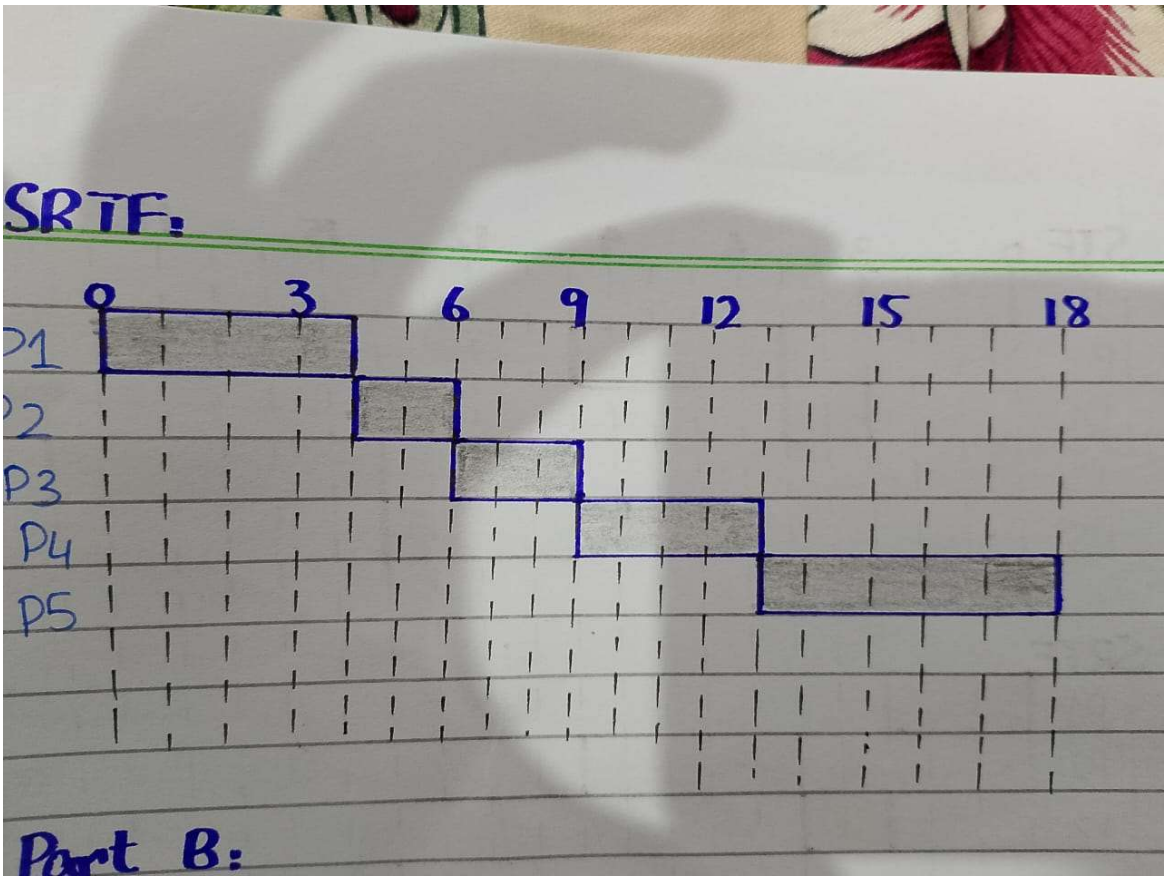
Part-A

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

Part A:

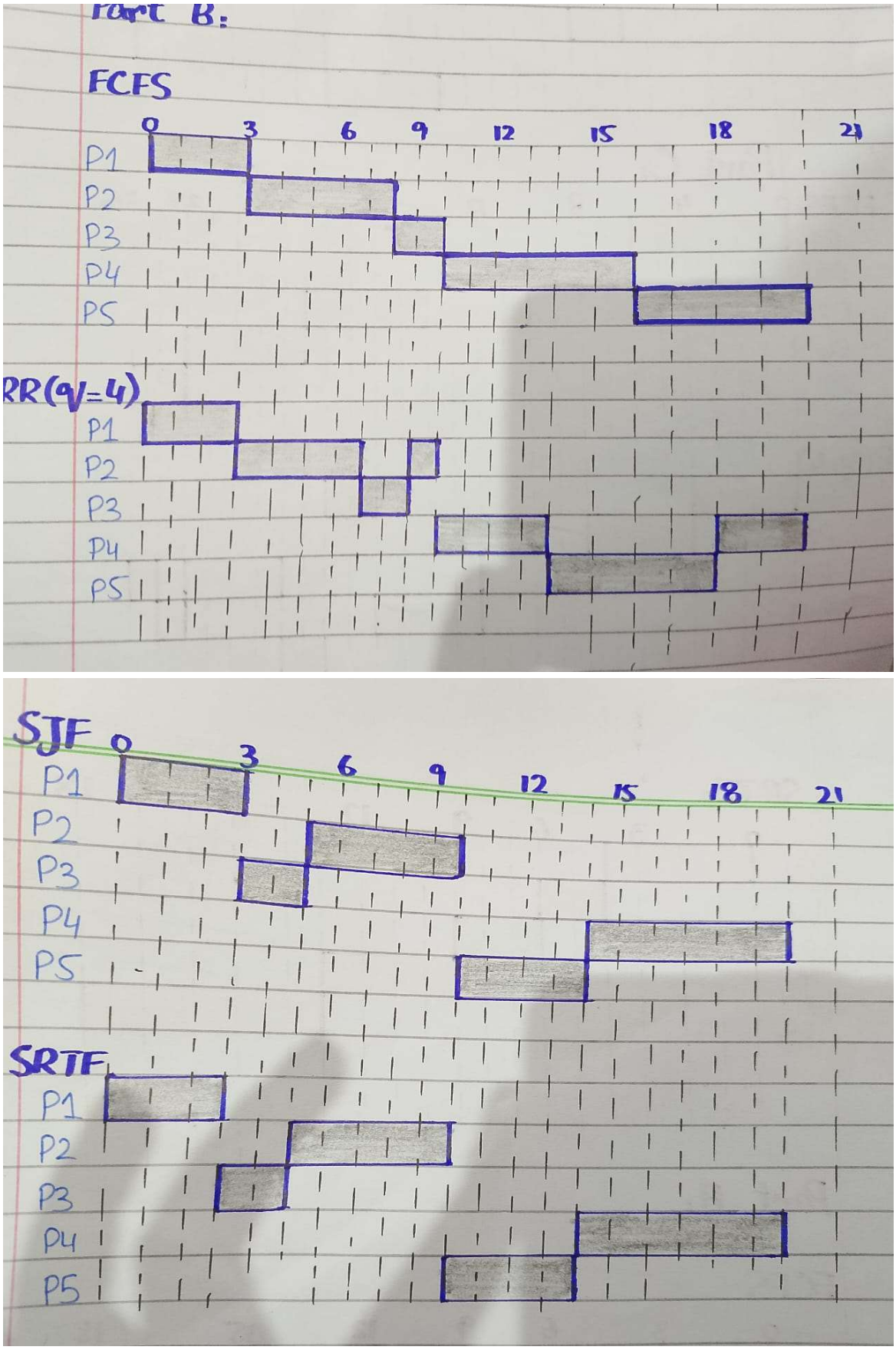
(a) Gantt Charts





Part-B

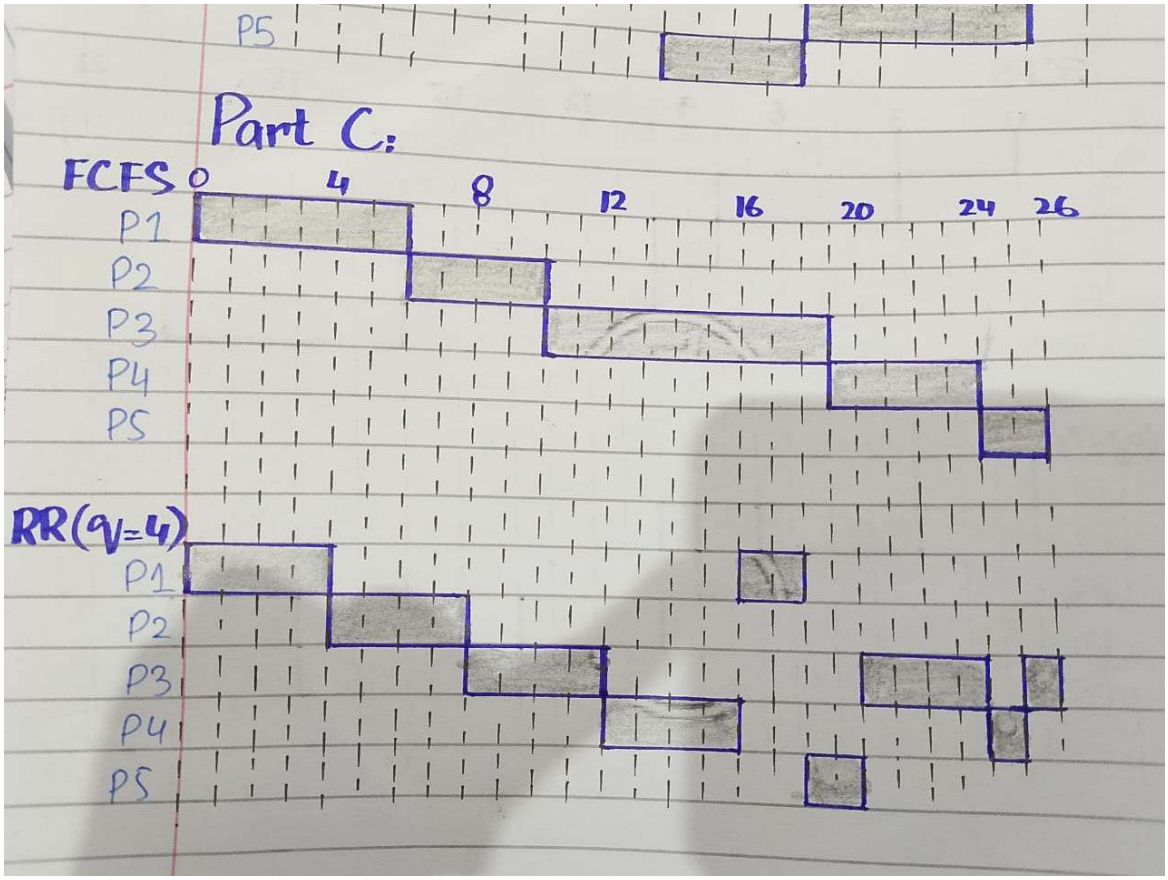
Process	Arrival Time	Service Time
P1	0	3
P2	1	5
P3	3	2
P4	9	6
P5	10	4

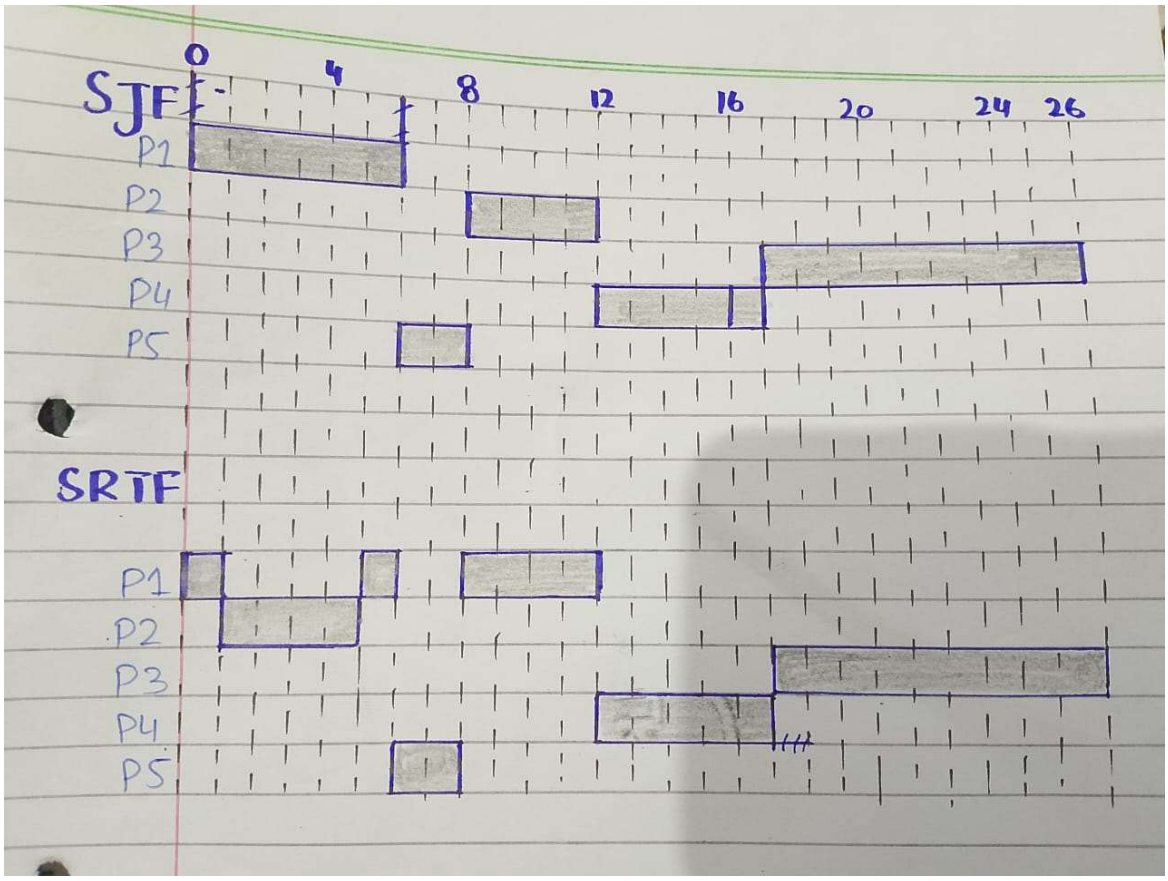


Part-C (Select Your own individual arrivals time and service time)

Process	Arrival Time	Service Time
P1	-	-

P2	-	-
P3	-	-
P4	-	-
P5	-	-





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

FCFS:

Process	Arrival Time	Service Time	Completion Time	Waiting Time	TurnAround Time(TAT)	TR/TS	CPU Idle Time
P1	0	4	6	0	6	1.00	0
P2	2	5	10	5	9	2.25	0
P3	4	2	19	8	17	1.89	0
P4	6	3	24	16	21	4.20	0
P5	9	4	26	18	20	10.00	0

- Avg waiting time = (0 + 5 + 8 + 16 + 18)/5 = 9.4
- Avg turnaround time = (6 + 9 + 17 + 21 + 20)/5 = 14.6
- CPU idle time = 0

RR(q=4):

Process	Arrival Time	Service Time	Completion Time	Waiting Time	TurnAround Time(TAT)	TR/TS	CPU Idle Time
P1	0	4	18	12	18	3.00	0
P2	2	5	8	3	7	1.75	0
P3	4	2	26	15	24	2.67	0
P4	6	3	25	17	22	4.40	0
P5	9	4	20	12	14	7.00	0

- Avg waiting time = (12+3+15+17+12)/5 = 11.8
- Avg turnaround time = (18+7+24+22+14)/5 = 17.0
- CPU idle time = 0

SJF:

Process	Arrival Time	Service Time	Completion Time	Waiting Time	TurnAround Time(TAT)	TR/TS	CPU Idle Time
P1	0	4	6	0	6	1.00	0
P2	2	5	12	0	2	1.00	0
P3	4	2	26	7	11	2.75	0
P4	6	3	17	9	14	2.80	0
P5	9	4	8	15	24	2.67	0

- Avg waiting time = (0+0+7+9+15)/5 = 6.2
- Avg turnaround time = (6+2+11+14+24)/5 = 11.4
- CPU idle time = 0

SRTF:

Process	Arrival Time	Service Time	Completion Time	Waiting Time	TurnAround Time(TAT)	TR/TS	CPU Idle Time
P1	0	4	12	6	12	2.00	0
P2	2	5	5	0	4	1.00	0
P3	4	2	26	15	24	2.67	0
P4	6	3	17	9	14	2.80	0
P5	9	4	8	0	2	1.00	0

- Avg waiting time = (6+0+15+9+0)/5 = 6.0
- Avg turnaround time = (12+4+24+14+2)/5 = 11.2
- CPU idle time = 0

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

Algorithm	Average Waiting Time	Average Turnaround Time
FCFS	9.4	14.6
RR(q=4)	11.8	17.0
SJF	6.2	11.4
SRTF	6.0	11.2

Submission Guidelines

- Submit a single PDF file on MS Teams including:

  – Screenshots of code and execution for all programming tasks. –
  Answers to all theory and analytical questions.

- Push all C source files and the PDF to your GitHub repository.
- Late submissions will not be accepted.

- Direct copied from any source will be penalized
- Deadline: 26th October 2025, 11:59 PM.

- VIVA will be held in coming week (week-7)