



DEPARTMENT OF COMPUTER SCIENCE

SUBMITTED BY:

Farah Naz

23-NTU-CS-1152

SECTION SE: 5th(A)

LAB MANUAL

SUBMITTED TO:

Sir Nasir Mehmood

SUBMISSION DATE:

10-26-2025

National Textile University, Faisalabad

Department of Computer Science

Course: Operating Systems (CSC-3075)

Semester: Fall 2025

BSSE 5th A

Instructor: Mr. Nasir Mahmood

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.

Code:

Output:

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.

Code:

Output:

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.”

Code:

The screenshot shows a code editor with a file explorer on the left. The file explorer lists files under 'ASSIGNMENT-1-1152 [WSL: UBUNTU-24.04]': question, question1, question1.c, question2, question2.c, question3, question3.c (selected), question4, question4.c, question5, and question5.c. The main editor window displays the code for 'question3.c'.

```
1 |
2 | #include <stdio.h>
3 | #include <pthread.h>
4 | void* print_number(void* arg) {
5 |
6 |     int num = *(int*)arg; // Cast void* back to int*
7 |     printf("Thread received number: %d\n", num);
8 |     printf("Square: %d\n", num * num);
9 |     printf("Cube: %d\n", num * num * num);
10 |    return NULL;
11 | }
12 | int main() {
13 |     pthread_t thread_id;
14 |     int number = 42;
15 |     printf("Creating thread with argument: %d\n", number);
16 |     pthread_create(&thread_id, NULL, print_number, &number);
17 |     pthread_join(thread_id, NULL);
18 |     printf("Main thread done.\n");
19 |     return 0;
20 | }
21 |
```

The status bar at the bottom indicates 'Ln 1, Col 1', 'Spaces: 4', 'UTF-8', 'LF', and the system tray shows '20°C' and '11:03 PM 10/26/2025'.

Output:

The screenshot shows a terminal window with the following output:

```
farah@DESKTOP-LI8S698:~/Assignment-1-1152$ ./question3
● farah@DESKTOP-LI8S698:~/Assignment-1-1152$ ./question3
Creating thread with argument: 42
Thread received number: 42
Square: 1764
Cube: 74088
Main thread done.
○ farah@DESKTOP-LI8S698:~/Assignment-1-1152$
```

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.

Code:

The screenshot shows the Visual Studio Code editor with a C program named `question4.c` open. The program includes `stdio.h`, `pthread.h`, and `stdlib.h`. It defines a function `comp_factorial` that takes a `void*` argument, casts it to an `int*`, and calculates the factorial of the number. The `main` function creates a thread to call `comp_factorial` with the number 3, prints the result, and then prints "Main thread done." before returning 0. The Explorer sidebar on the left shows a project named "ASSIGNMENT-1-1152 [WSL: UBUNTU-24.04]" with files `question1.c` through `question5.c`. The bottom status bar indicates the file is at line 15, column 25, with 4 spaces, UTF-8 encoding, and LF line endings.

```
1 #include <stdio.h>
2 #include <pthread.h>
3 #include <stdlib.h>
4 void* comp_factorial(void* arg) {
5
6     int num = *(int*)arg; // Cast void* back to int*
7     int* factorial= malloc(sizeof(int));
8     *factorial=1;
9     for(int i=num;i>0;i--){
10         *factorial*=i;
11     }
12     return (void*)factorial;
13 }
14 int main() {
15     pthread_t thread_id;
16     int number = 3;
17     void* factorial;
18     printf("The number given is: %d\n", number);
19     pthread_create(&thread_id, NULL, comp_factorial, &number);
20     pthread_join(thread_id, &factorial);
21     printf("The factorial that we received is: %d\n",*(int*)factorial);
22     printf("Main thread done.\n");
23     return 0;
24 }
```

Output:

The screenshot shows a terminal window with the following output:

```
farah@DESKTOP-LI8S698:~/Assignment-1-1152$ gcc question3.c -o question3
farah@DESKTOP-LI8S698:~/Assignment-1-1152$ ./question3
The number given is: 3
The factorial that we received is: 6
Main thread done.
farah@DESKTOP-LI8S698:~/Assignment-1-1152$
```

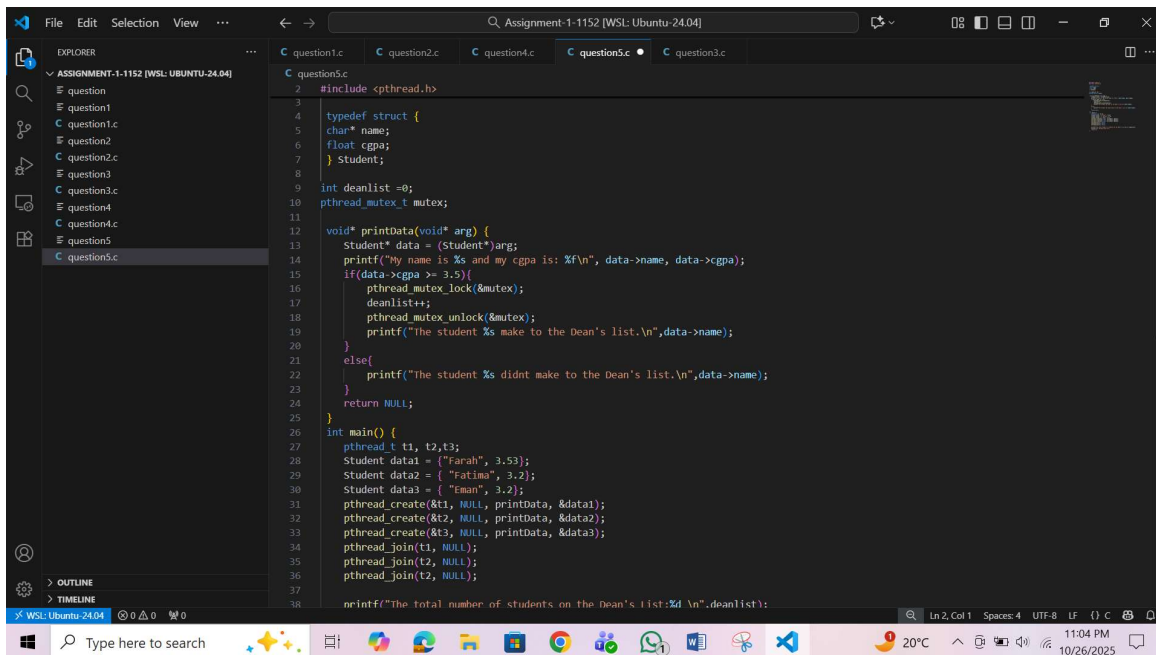
The bottom right corner of the terminal window shows "Ln 25, Col 1".

Task 5 – Struct-Based Thread Communication

Create a program that simulates a simple student database system.

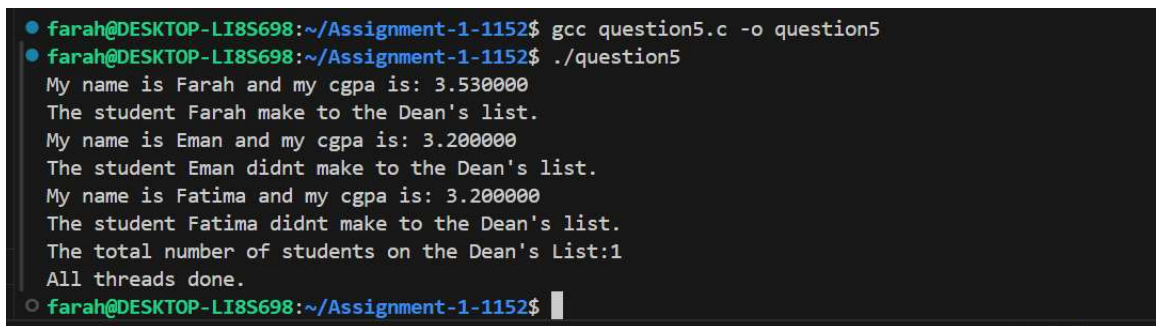
- Define a struct: `typedef struct { int student_id; char name[50]; float gpa; } Student;`
- Create 3 threads, each receiving a different Student struct.
- Each thread prints student info and checks Dean's List eligibility ($GPA \geq 3.5$).
- The main thread counts how many students made the Dean's List.

Code:



```
1 #include <pthread.h>
2
3 typedef struct {
4     char* name;
5     float cgpa;
6 } Student;
7
8 int deanlist = 0;
9 pthread_mutex_t mutex;
10
11 void* printData(void* arg) {
12     Student* data = (Student*)arg;
13     printf("My name is %s and my cgpa is: %f\n", data->name, data->cgpa);
14     if(data->cgpa >= 3.5){
15         pthread_mutex_lock(&mutex);
16         deanlist++;
17         pthread_mutex_unlock(&mutex);
18         printf("The student %s make to the Dean's list.\n", data->name);
19     }
20     else{
21         printf("The student %s didnt make to the Dean's list.\n", data->name);
22     }
23     return NULL;
24 }
25
26 int main() {
27     pthread_t t1, t2, t3;
28     Student data1 = {"Farah", 3.53};
29     Student data2 = {"Fatima", 3.2};
30     Student data3 = {"Eman", 3.2};
31     pthread_create(&t1, NULL, printData, &data1);
32     pthread_create(&t2, NULL, printData, &data2);
33     pthread_create(&t3, NULL, printData, &data3);
34     pthread_join(t1, NULL);
35     pthread_join(t2, NULL);
36     pthread_join(t3, NULL);
37
38     printf("The total number of students on the Dean's list: %d\n", deanlist);
39 }
```

Output:



```
farah@DESKTOP-LI8S698:~/Assignment-1-1152$ gcc question5.c -o question5
farah@DESKTOP-LI8S698:~/Assignment-1-1152$ ./question5
My name is Farah and my cgpa is: 3.530000
The student Farah make to the Dean's list.
My name is Eman and my cgpa is: 3.200000
The student Eman didnt make to the Dean's list.
My name is Fatima and my cgpa is: 3.200000
The student Fatima didnt make to the Dean's list.
The total number of students on the Dean's List:1
All threads done.
farah@DESKTOP-LI8S698:~/Assignment-1-1152$
```

Section-B: Short Questions

1. Answer all questions briefly and clearly.
2. **Define an Operating System in a single line.**

Answer:

An operating system (OS) is system software that manages a computer's hardware and software resources to perform tasks, acting as an intermediary between the user and the computer.

3. **What is the primary function of the CPU scheduler?**

Answer:

CPU scheduling is the process of deciding which process will own the CPU to use while another process is suspended. The main function of CPU scheduling is to ensure that whenever the CPU remains idle, the OS has at least selected one of the processes available in the ready-to-use line.

4. **List any three states of a process.**

Answer:

The three states of process are:

1. Ready.
2. Running.
3. Waiting.

5. **What is meant by a Process Control Block (PCB)?**

Answer:

A Process Control Block is a data structure used by an operating system to manage and control processes. It holds all the necessary information about a process, such as its state, unique process ID, program counter, CPU registers, memory management details, allowing the OS to perform tasks like multitasking and context switching.

6. **Differentiate between a process and a program.**

Answer:

Program: A program is a passive set of instructions stored on disk, like a software application file

Process: A process is an active, running instance of that program, loaded into memory and utilizing system resources like CPU and RAM to perform its tasks

7. **What do you understand by context switching?**

Answer:

Context switching is a process in an operating system where OS stops the execution of one program in running state and loads another program by loading state of 2nd program. This switching of programs allows multitasking where it seems that all programs are executing at once.

8. **Define CPU utilization and throughput.**

Answer:

CPU Utilization: It is the total percentage of the time the processor is busy executing tasks.

Throughput: it is the number of tasks completed per unit time.

9. **What is the turnaround time of a process?**

Answer:

Turnaround Time:

For a particular process, the important conditions are how long it takes to perform that process. The time elapsed from the time of process delivery to the time of completion is known as the conversion time. Conversion time is the amount of time spent waiting for memory access, waiting in line, using CPU and waiting for I/O.

10. **How is waiting time calculated in process scheduling?**

Answer:

Waiting time = Turnaround time – Service time.

11. Define response time in CPU scheduling.

Answer:

Response time is the time taken in the submission of the application process until the first response is issued.

12. What is preemptive scheduling?

Answer:

Preemptive scheduling is used when a process switches from running state to ready state or from the waiting state to the ready state.

13. What is non-preemptive scheduling?

Answer:

Non-Preemptive scheduling is used when a process terminates , or when a process switches from running state to waiting state.

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

Answer:

- Fairness: Each process gets an equal share of the CPU.
- Simplicity: The algorithm is straightforward and easy to implement.

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

Answer:

- SJF may cause very long turn-around times or starvation.

16. Define CPU idle time.

Answer:

The time during which the CPU remains unused because no process is ready to execute.

17. State two common goals of CPU scheduling algorithms.

Answer:

- Maximize CPU utilization.
- Minimize waiting and turnaround times.

18. List two possible reasons for process termination.

Answer:

- Process completes execution.
- Process is killed by the user or operating system.

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

Answer:

- `wait()`: Makes a parent process wait until its child finishes.
- `exit()`: Terminates the calling process and returns a status to the parent.

20. Differentiate between shared memory and message-passing models of inter-process communication.

Answer:

- **Shared Memory**: Processes communicate through a common memory region.
- **Message Passing**: Processes communicate by sending and receiving messages.

21. Differentiate between a thread and a process.

Answer:

- **Process**: Independent execution having its own memory space.
- **Thread**: A lightweight unit within a process sharing the same memory as the process.

22. Define multithreading.

Answer:

- **Multithreading** is the ability of a CPU to execute multiple threads, or sequences of instructions, from a single program, improving performance and responsiveness.

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

Answer:

- A **CPU-bound process** is one that spends most of its time performing calculations and is limited by the CPU's speed.
- **I/O-bound process** is limited by the speed of input/output devices and spends most of its time waiting for operations like reading from or writing to a disk, network, or database to complete.

24. What are the main responsibilities of the dispatcher?

Answer:

- **Context switching**: The dispatcher saves the state (or context) of the process that was just running and loads the saved context of the next

process. The context includes all the data required to restart a process, such as the CPU registers, program counter, and memory management information.

- **Switching to user mode:** The dispatcher is responsible for switching the CPU from kernel mode to user mode before handing control to the selected process.

25. **Define starvation and aging in process scheduling.**

Answer:

- **Starvation:** A process waits indefinitely for CPU.
- **Aging:** Gradually increasing the priority of waiting processes to prevent starvation.

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

Answer:

The maximum amount of time a process is allowed to run on a CPU in a preemptive multitasking system before the operating system switches to another process.

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

Answer:

1. If the time quantum is too large

- **Behaves like FCFS:** The scheduling behavior becomes similar to FCFS because processes are likely to run for their entire burst time before another process gets a turn.
- **Poor response time:** Interactive processes will have a very long wait time to get a response, making the system feel sluggish.

2. If the time quantum is too small

- **High overhead:** The system spends a significant amount of time on context switching between processes.
- **Low throughput:** The amount of actual work done decreases because the CPU spends less time executing processes and more time on the overhead of switching tasks.

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

Answer:

In an operating system, the turnaround ratio is a performance metric that compares the total time a process spends in the system to the time it spends actually executing. It is also known as the normalized turnaround time.

29. **What is the purpose of a ready queue?**

Answer:

The purpose of a ready queue is to hold all processes that are in main memory and are ready to be executed by the CPU. It serves as a central waiting list from which the CPU scheduler selects the next process to run when a CPU becomes available.

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

Answer:

- A CPU burst is a period of time a process uses the CPU to execute instructions. CPU bursts involve active computation and keep the CPU busy.
- I/O burst is the time a process spends waiting for an input/output operation to complete., whereas I/O bursts involve a process being idle and waiting for external devices like a disk or network to finish.

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

Answer:

Algorithm: Round Robin.

Reason: Because each process gets equal CPU time in cyclic order.

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

Answer:

1. **Parent** process calls **fork()** to create a child.
2. **Child** gets a copy of the parent's address space.
3. Child may call **exec()** to load a new program.
4. Parent may use **wait()** for child termination.

33. **Define zombie and orphan processes.**

Answer:

- A **zombie** process has completed execution but its entry remains in the process table because the parent hasn't read its exit status.
- An **orphan** process is a running process whose parent has terminated before it, and it is then adopted by the **init** process (PID 1).

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

Answer:

Priority Scheduling	Shortest Job First
---------------------	--------------------

1. Priority scheduling selects the process with the highest priority	SJF selects the process with the smallest estimated execution time. SJF is considered a special case of priority scheduling where the priority is based solely on the burst time.
2. Priority scheduling can be based on a static or dynamic priority value assigned to each process	SJF uses the process's execution length as its scheduling criterion.

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

Answer:

- **Context switch** time is the duration required for a CPU to switch from executing one process to another by saving the first process's state and loading the second.
- It is considered overhead because the CPU performs no useful user-level work during this time; it is simply executing internal system tasks to manage the transition.

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

Answer:

- **Long-term Scheduler (Job Scheduler):** The long-term scheduler selects processes from the "job pool" (secondary storage) and loads them into the main memory to be ready for execution.
- **Medium-term Scheduler (Swapping Scheduler):** The medium-term scheduler is responsible for swapping processes in and out of main memory. When memory becomes overcommitted, it can remove a process from memory, storing it in secondary storage.
- **Short-term Scheduler (CPU Scheduler):** This is the most frequent scheduler, deciding which process from the "ready queue" (processes in main memory ready to run) should be assigned to the CPU next.

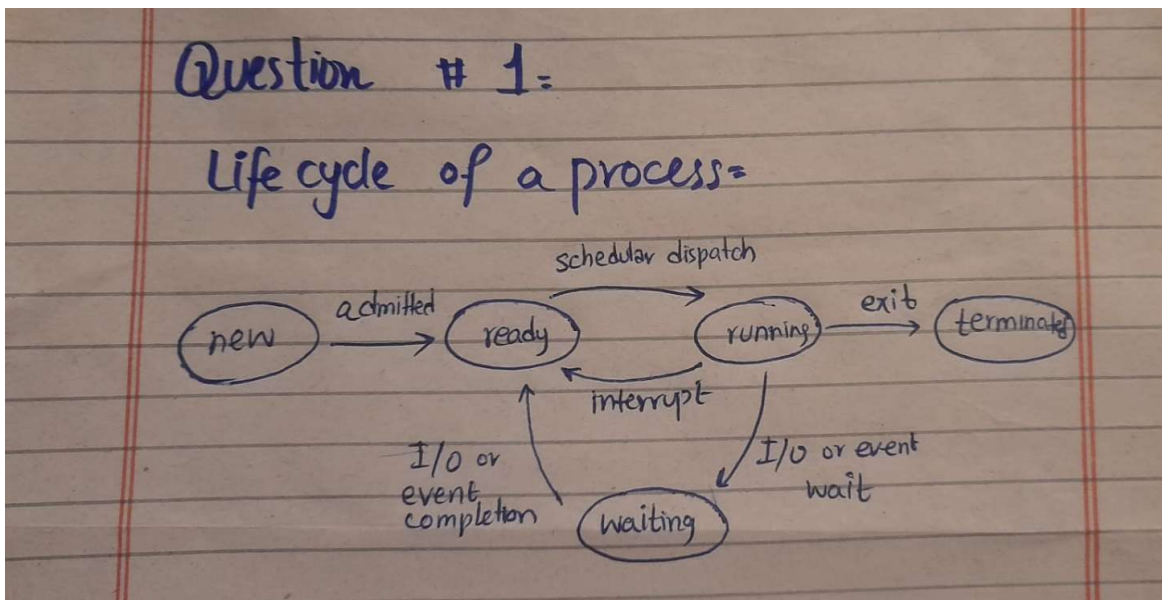
37. **Differentiate between User Mode and Kernel Mode in an Operating System.**

Answer:

Feature	User Mode	Kernel Mode
Privilege Level	Low (unprivileged)	High (privileged)
Access to Resources	Restricted; limited access to hardware and memory	Full access to all system resources, hardware, and memory
Primary Use	Running applications like web browsers and word processors	Core OS functions like memory management, process scheduling, and device management
Memory Management	Each process has a separate, isolated virtual address space	All processes share a single virtual address space

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.**

Answer:

Context Switch:

Context switching is the process where the CPU stops running one process, saves its current state, and loads the saved state of another process so that multiple processes can share the CPU effectively.

The information saved and restored includes:

- **Program Counter (PC):** Also known as the Instruction Pointer, this register holds the address of the next instruction to be executed. Saving it is crucial for knowing where to restart the process.
- **CPU registers:** These include all the general-purpose registers that the process was using, such as data registers, address registers, and special-purpose registers for floating-point or vector operations.
- **Process state:** The current status of the process (e.g., Running, Ready, Waiting, Terminated) is saved to the PCB.
- **CPU scheduling information:** This includes the process's priority, pointers to its location in various queues (e.g., the ready queue), and other algorithm-specific parameters.
- **Memory management information:** To manage the process's address space, the operating system must save information like page tables or segment tables.
- **I/O status information:** A list of any I/O devices allocated to the process and a list of its open files are saved.

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

Answer:

PCB:

A Process Control Block (PCB) is a data structure used by the operating system to manage information about a process. The process control keeps track of many important pieces of information needed to manage processes efficiently.

Components of PCB:

- **Pointer:** It is a stack pointer that is required to be saved when the process is switched from one state to another to retain the current position of the process.
- **Process state:** It stores the respective state of the process.
- **Process number:** Every process is assigned a unique id known as process ID or PID which stores the process identifier.

- **Program counter:** Program Counter stores the counter, which contains the address of the next instruction that is to be executed for the process.
- **Register:** Registers in the PCB, it is a data structure. When a process is running and its time slice expires, the current value of process specific registers would be stored in the PCB and the process would be swapped out. When the process is scheduled to be run, the register values are read from the PCB and written to the CPU registers. This is the main purpose of the registers in the PCB.
- **Memory limits:** This field contains the information about memory management system used by the operating system. This may include page tables, segment tables, etc.
- **List of Open files:** This information includes the list of files opened for a process.

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

Answer:

Basis	Short-Term Scheduler	Medium-term Scheduler	Long-Term Scheduler
1. Alternate Name	It is also called a CPU scheduler.	It is also called a process swapping scheduler.	It is also called a job scheduler.
2. Degree in programming	It provides lesser control over the degree of multiprogramming.	It reduces the control over the degree of multiprogramming.	It controls the degree of multiprogramming.
3. Speed	The speed of the short-term scheduler is very fast.	Speed of medium scheduler is between the short-term and long-term scheduler.	The speed of a long-term scheduler is more than medium-term scheduler.
5. Purpose	It selects the processes from among the process	It can reintroduce the from among the process into	It selects processes from the pool and

Basis	Short-Term Scheduler	Medium-term Scheduler	Long-Term Scheduler
	that is ready to execute.	memory that executes and its execution can be continued.	loads them into memory for execution.
6. Process state	Process state is ready to running	Process state is not present	Process state is new to ready.
7. Selection of process	Select a new process for a CPU quite frequently.	Select that process, which is currently not need to load fully on RAM, so it swap it into swap partition.	Select a good process, mix of I/O bound and CPU bound.

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

Answer:

CPU Scheduling Criteria:

- **CPU utilization:** The main objective of any CPU scheduling algorithm is to keep the CPU as busy as possible. Theoretically, CPU utilization can range from 0 to 100 but in a real-time system, it varies from 40 to 90 percent depending on the load upon the system.
- **Throughput:** A measure of the work done by the CPU is the number of processes being executed and completed per unit of time. This is called throughput. The throughput may vary depending on the length or duration of the processes.
- **Turnaround Time:** For a particular process, an important criterion is how long it takes to execute that process. The time elapsed from the time of submission of a process to the time of completion is known as the turnaround time. Turn-around time is the sum of times spent waiting to get into memory, waiting in the ready queue, executing in CPU and waiting for I/O.

- **Waiting Time:** A scheduling algorithm does not affect the time required to complete the process once it starts execution. It only affects the waiting time of a process i.e. time spent by a process waiting in the ready queue.

$$\text{Waiting Time} = \text{Turnaround Time} - \text{Burst Time.}$$

- **Response Time:** A process may produce some output fairly early and continue computing new results while previous results are being output to the user. Thus, another criterion is the time taken from submission of the process of the request until the first response is produced. This measure is called 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.
- 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

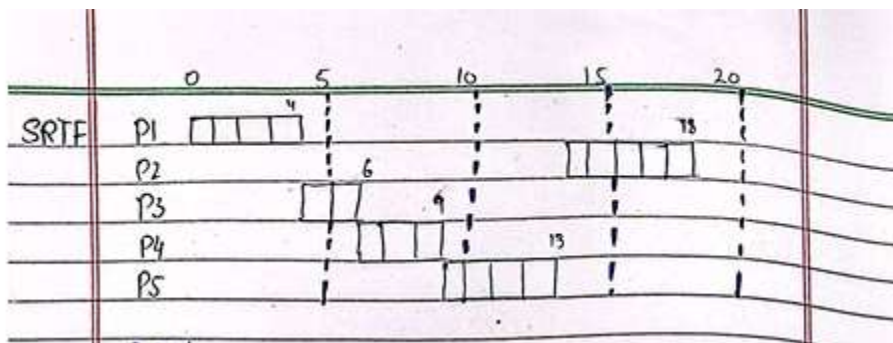
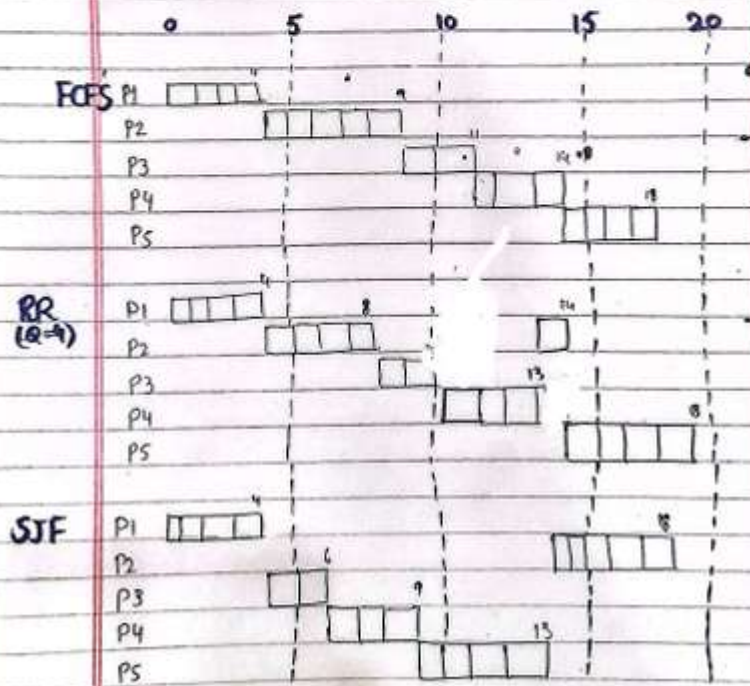
Process	Arrival Time	Service Time
P1	0	4
P2	2	5
P3	4	2
P4	6	3
P5	9	4

Section - D:- CPU Scheduling Cal

Part A

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

Gantt Chart



FCFS =

Process	Finish	Turnaround	Service Time	Waiting	Tr/Ts
P1	4	4	4	0	1.0
P2	9	7	5	2	1.4
P3	11	7	2	5	3.5
P4	14	8	3	5	2.67
P5	18	9	4	5	2.25

- Avg waiting time = 3.4
- Avg turnaround time = 7
- Avg TR/TS Ratio = 2.16
- CPU Idle Time = 0

Round Robin ($Q=4$).

Process	Finish	Turnaround	Service Time	Waiting
P1	4	4	4	0
P2	18	12	5	7
P3	10	6	2	4
P4	13	7	3	4
P5	17	8	4	4

• Avg waiting Time = 3.8

• Avg turnaround time = 7.4

• Avg $T_r/T_s = 2.0$

• CPU Idle Time = 0

SJF Analysis =

Process	Finish	Turnaround	Service Time	Waiting
P1	4	4	4	0
P2	18	16	5	11
P3	6	2	2	0
P4	9	3	3	0
P5	13	4	4	0

• Avg waiting time = 2.2

• Avg Turnaround time = 5.8

• Avg $T_r/T_s = 1.44$

• CPU Idle Time = 0

SRTF Analysis =

Process	Finish Time	Turnaround Time	Service Time	Waiting Time	Tr/Ts
P1	4	4	4	0	1.0
P2	18	16	5	11	3.2
P3	6	2	2	0	1.0
P4	9	3	3	0	1.0
P5	13	4	4	0	1.0

• Avg waiting time = 2.2

• Avg Turnaround time = 5.8

• Avg Tr/Ts time = 1.44

• CPU Idle Time = 0

Best Algorithm =

• SRTF and SJF are best performing for this dataset

• Lowest average waiting and turnaround time

• Lowest average Tr/Ts time as compared to others.

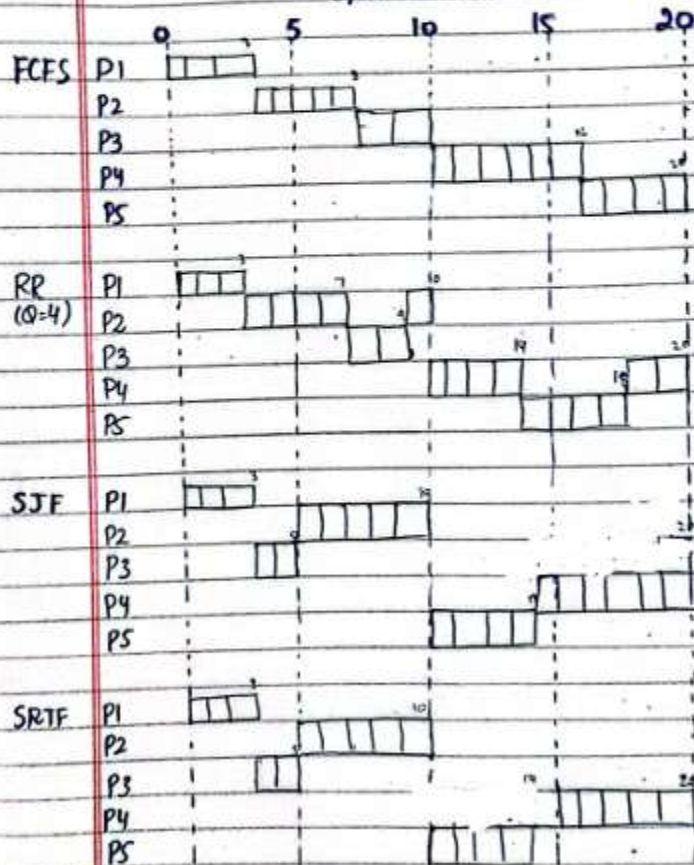
Part-B

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

Part B:-

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

Gantt Chart



FCFS Analysis =

Process	Finish	T_r	T_s	Waiting	T_r/T_s
P1	3	3	3	0	1
P2	8	7	5	2	1.4
P3	10	7	2	5	3.5
P4	16	7	6	1	1.16
P5	20	10	4	6	2.5

• Avg waiting time = 2.8

• Avg turnaround time = 6.8

• Avg T_r/T_s time = 1.91

• CPU Idle time = 0

Round Robin (Q=4) Analysis =

Process	Finish	T_r	T_s	Waiting	T_r/T_s
P1	3	3	3	0	1.0
P2	10	9	5	4	1.8
P3	9	6	2	4	3.0
P4	20	11	6	5	1.83
P5	18	8	4	4	2.0

• Avg waiting time = 3.4

• Avg turnaround time = 7.4

• Avg T_r/T_s time = 1.93

• CPU Idle time = 0

SIF Analysis.

Process	Finish	T_r	T_s	Waiting	T_r/T_s
P1	3	3	3	0	1.0
P2	10	9	5	4	1.8
P3	5	2	2	0	1.0
P4	20	11	6	5	1.83
P5	14	4	4	0	1.0

• Avg waiting time = 1.8

• Avg Turnaround time = 5.8

• Avg T_r/T_s ratio = 1.33

• CPU Idle time = 0

SRIF Analysis.

Process	Finish	T_r	T_s	Waiting	T_r/T_s
P1	3	3	3	0	1.0
P2	10	9	5	4	1.8
P3	5	2	2	0	1.0
P4	20	11	6	5	1.83
P5	14	4	4	0	1.0

• Avg waiting time = 1.8

• Avg turnaround time = 5.8

• Avg T_r/T_s ratio = 1.33

• CPU Idle time = 0

Best Algorithm.

• SRIF and SIF are best performing for this dataset.

• Lowest average waiting and turnaround time.

• T_r/T_s ratio is lowest as compared to others.

• Optimal for minimizing metrics.

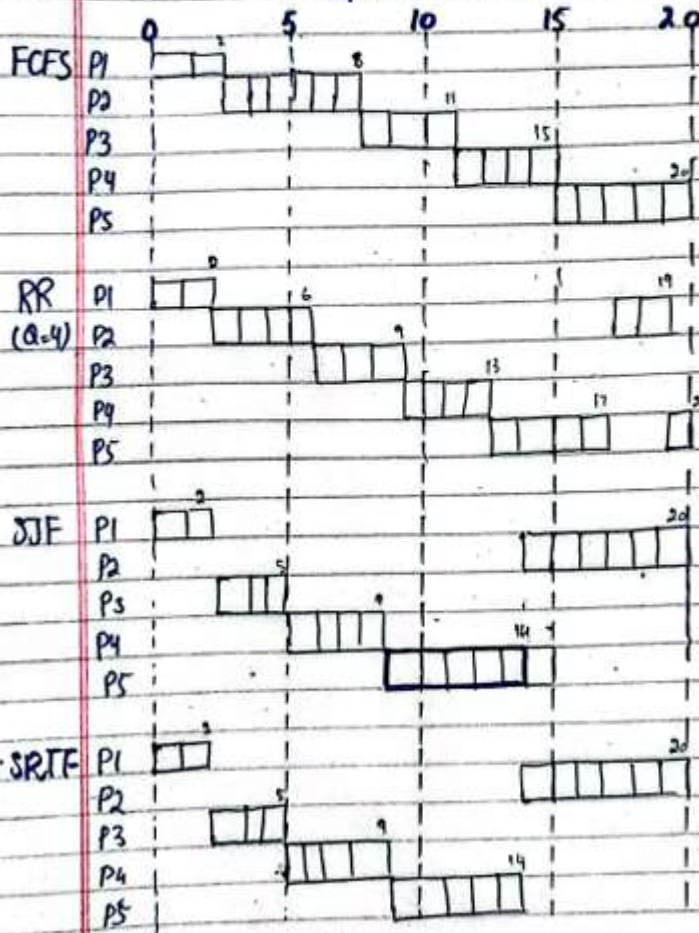
Part-C (Select Your own individual arrivals time and service time)

Process	Arrival Time	Service Time
P1	-	-
P2	-	-
P3	-	-
P4	-	-
P5	-	-

Part C:-

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

Gantt chart



FCFS Analysis =

Process	Finish	Tr	Ts	Waiting	Tr/Ts
P1	2	2	2	0	1
P2	8	7	6	1	1.16
P3	11	9	3	6	3.0
P4	15	12	4	8	3
P5	20	16	5	11	3.2

• Avg waiting time = 5.2

• Avg turnaround time = 9.2

• Avg tr/ts time = 2.27

• CPU idle time = 0

RR (Q=4) Analysis =

Process	Finish	Tr	Ts	Waiting	Tr/Ts
P1	2	2	2	0	1.0
P2	19	18	6	12	3.0
P3	9	7	3	4	2.3
P4	13	10	4	6	2.5
P5	20	16	5	11	3.2

• Avg waiting time = 6.6

• Avg turnaround time = 10.6

• Avg tr/ts time = 2.4

• CPU idle time = 0

SJF Analysis =

Process	Finish	T_r	T_s	Waiting	T_r/T_s
P1	2	2	2	0	1.0
P2	20	19	6	13	3.16
P3	5	3	3	0	1.0
P4	9	6	4	2	1.5
P5	14	9	5	4	1.8

• Avg waiting time = 3.8

• Avg turnaround time = 7.8

• Avg T_r/T_s time = 1.692

• CPU Idle time = 0

SRTF Analysis =

Process	Finish	T_r	T_s	Waiting	T_r/T_s
P1	2	2	2	0	1.0
P2	20	19	6	13	3.16
P3	5	3	3	0	1.0
P4	9	6	4	2	1.5
P5	14	9	5	4	1.8

• Avg waiting time = 3.8

• Avg turnaround time = 7.8

• Avg T_r/T_s time = 1.692

• CPU Idle time = 0

Best Algorithms:

- SRTF and SJF are best performing for this dataset.
- lowest average waiting and turnaround time
- T_r/T_s ratio is lowest as compared to others.
- Optimal for minimizing metrics

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 • VIVA will be held in coming week (week-7)
- Deadline: 26th October 2025, 11:59 PM.