Department of Computer Science

Subject:

OPERATING SYSTEM

Submitted by:

ABDUL REHMAN SUDAIS

Reg number:

23-NTU-CS-1123

Assignment no. :

1

Semester:

5TH

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.

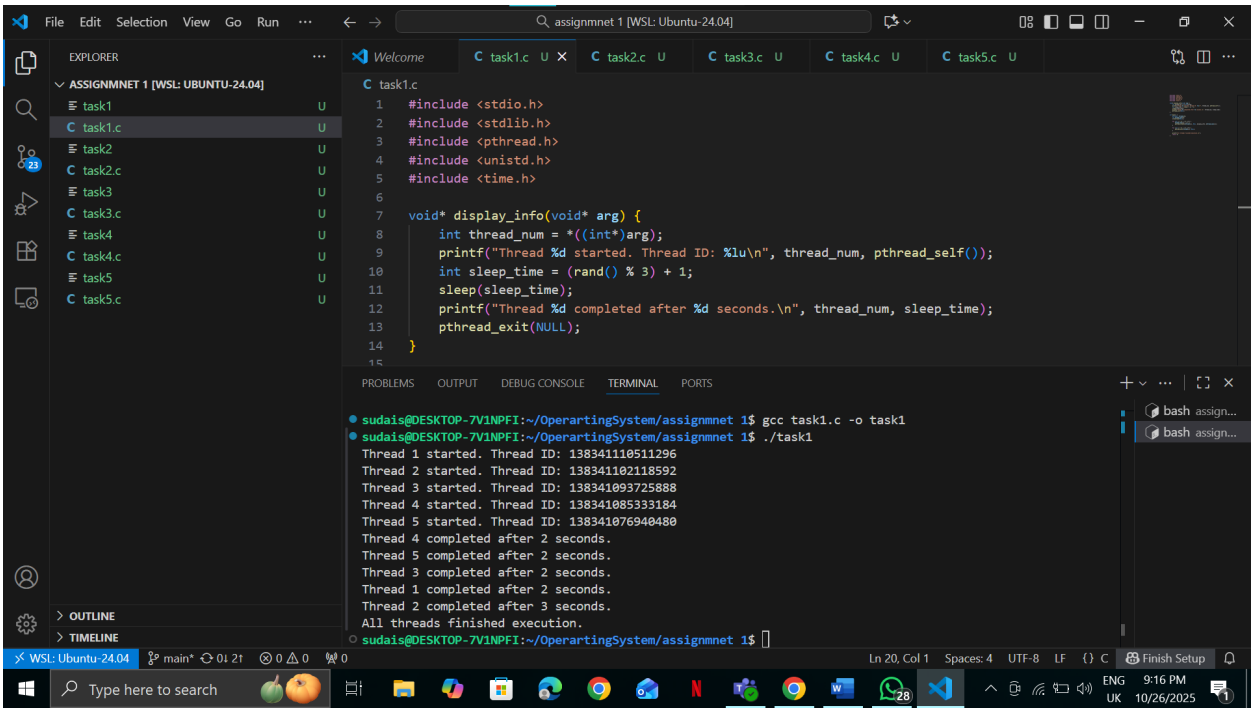
Expected Output: Threads complete in different orders due to random sleep times.

CODE:

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

void* display_info(void* arg) {
    int thread_num = *((int*)arg);
    printf("Thread %d started. Thread ID: %lu\n", thread_num, pthread_self());
    int sleep_time = (rand() % 3) + 1;
    sleep(sleep_time);
    printf("Thread %d completed after %d seconds.\n", thread_num, sleep_time);
    pthread_exit(NULL);
}

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

    for (int i = 0; i < 5; i++) {
```

```
        thread_nums[i] = i + 1;
        pthread_create(&threads[i], NULL, display_info, &thread_nums[i]);
    }

    for (int i = 0; i < 5; i++) {
        pthread_join(threads[i], NULL);
    }

    printf("All threads finished execution.\n");
    return 0;
}
```

- **OUTPUT:**



Task 2 – Personalized Greeting Thread

Write a C program that:

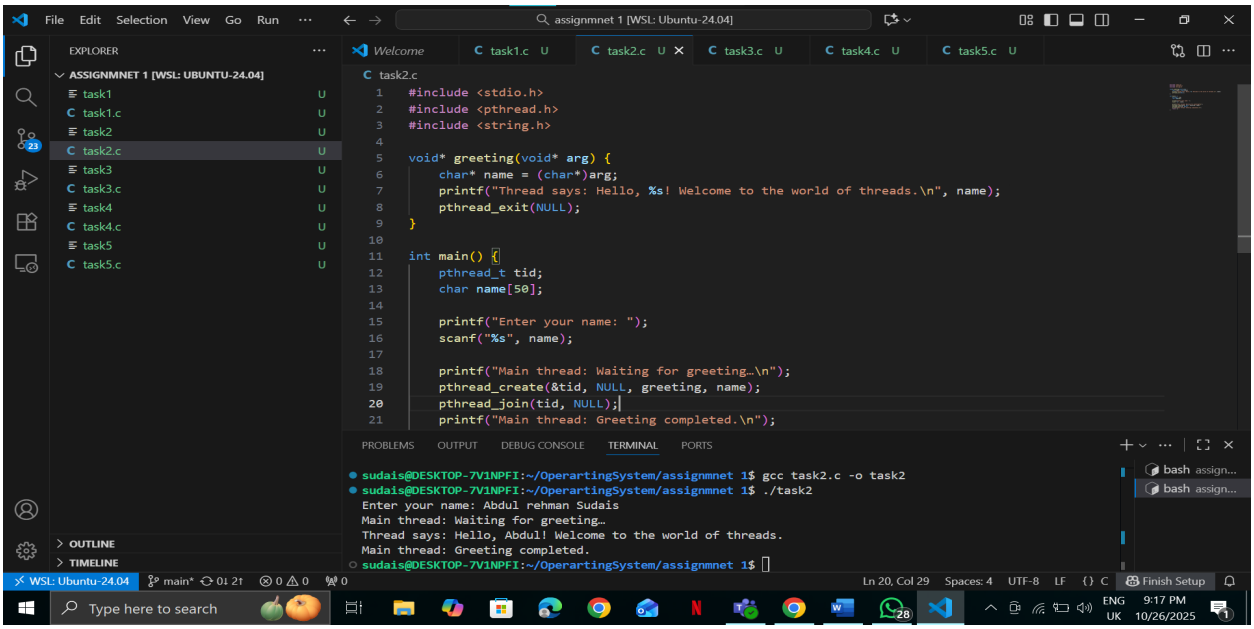
- Creates a thread that prints a personalized greeting message.
- The message includes the user's name passed as an argument to the thread.
- The main thread prints "Main thread: Waiting for greeting..." before joining the created thread.

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

CODE:

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

void* greeting(void* arg) {
    char* name = (char*)arg;
    printf("Thread says: Hello, %s! Welcome to the world of threads.\n", name);
    pthread_exit(NULL);
}

int main() {
    pthread_t tid;
    char name[50];

    printf("Enter your name: ");
    scanf("%s", name);

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

- **OUTPUT:**



Task 3 – Number Info Thread

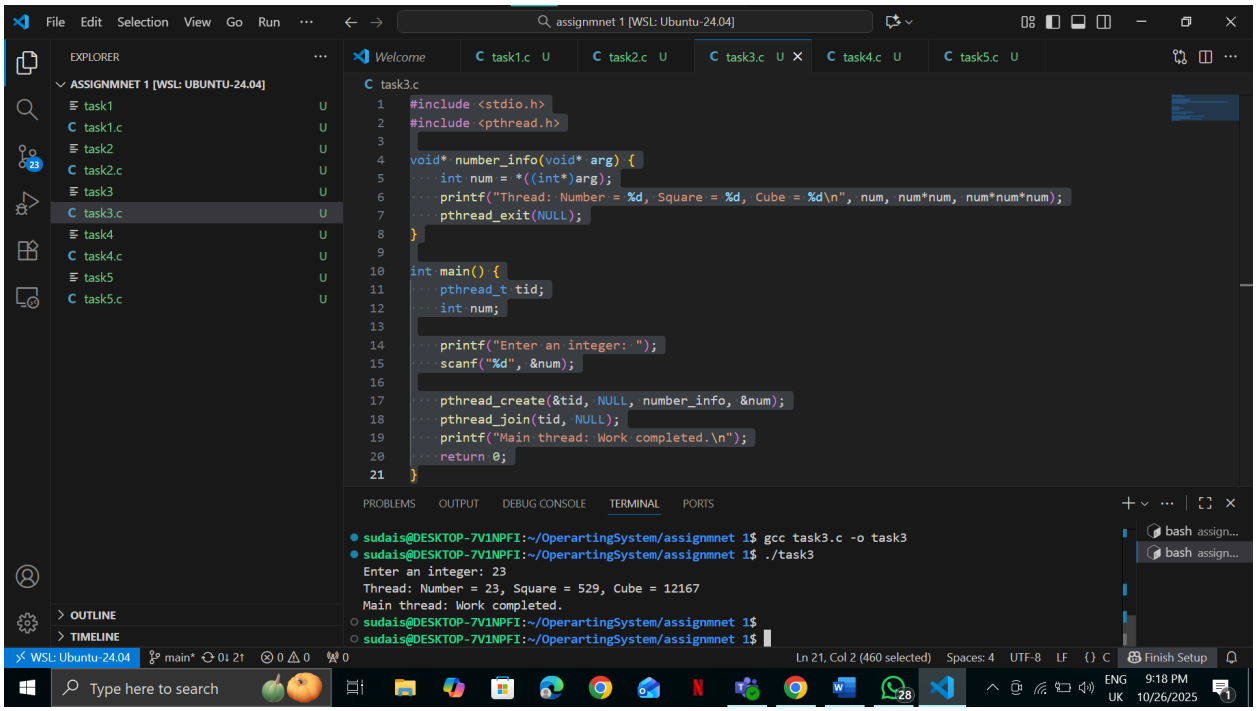
Write a program that:

• 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:

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

void* number_info(void* arg) {
    int num = *((int*)arg);
    printf("Thread: Number = %d, Square = %d, Cube = %d\n", num, num*num,
num*num*num);
    pthread_exit(NULL);
}

int main() {
    pthread_t tid;
    int num;

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

    pthread_create(&tid, NULL, number_info, &num);
    pthread_join(tid, NULL);
    printf("Main thread: Work completed.\n");
    return 0;
}
```

- **OUTPUT:**



Task 4 – Thread Return Values

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

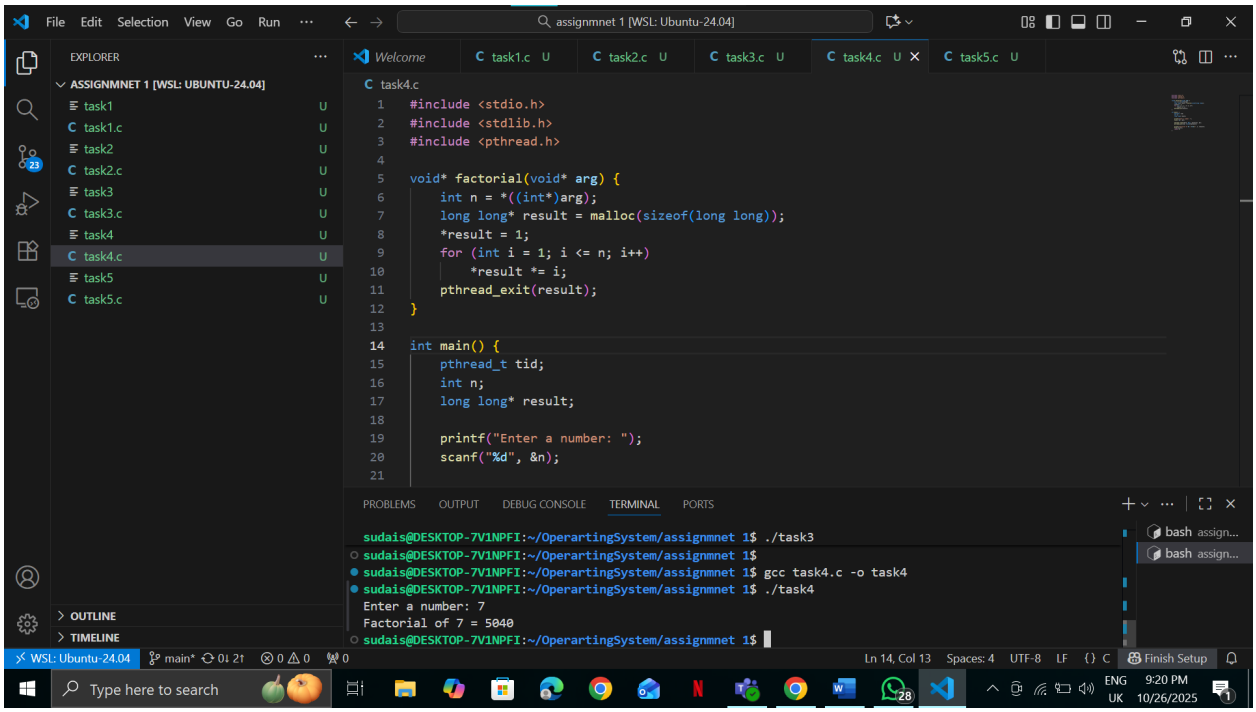
- The thread should return the result using a pointer.
- The main thread prints the result after joining.

CODE:

```c
#include <stdio.h>

#include <stdlib.h>
#include <pthread.h>

void* 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;
    pthread_exit(result);
}

int main() {
    pthread_t tid;
    int n;
    long long* result;
```

```c
    printf("Enter a number: ");
    scanf("%d", &n);

    pthread_create(&tid, NULL, factorial, &n);
    pthread_join(tid, (void**)&result);

    printf("Factorial of %d = %lld\n", n, *result);
    free(result);
    return 0;
}
```

- **OUTPUT:**



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.

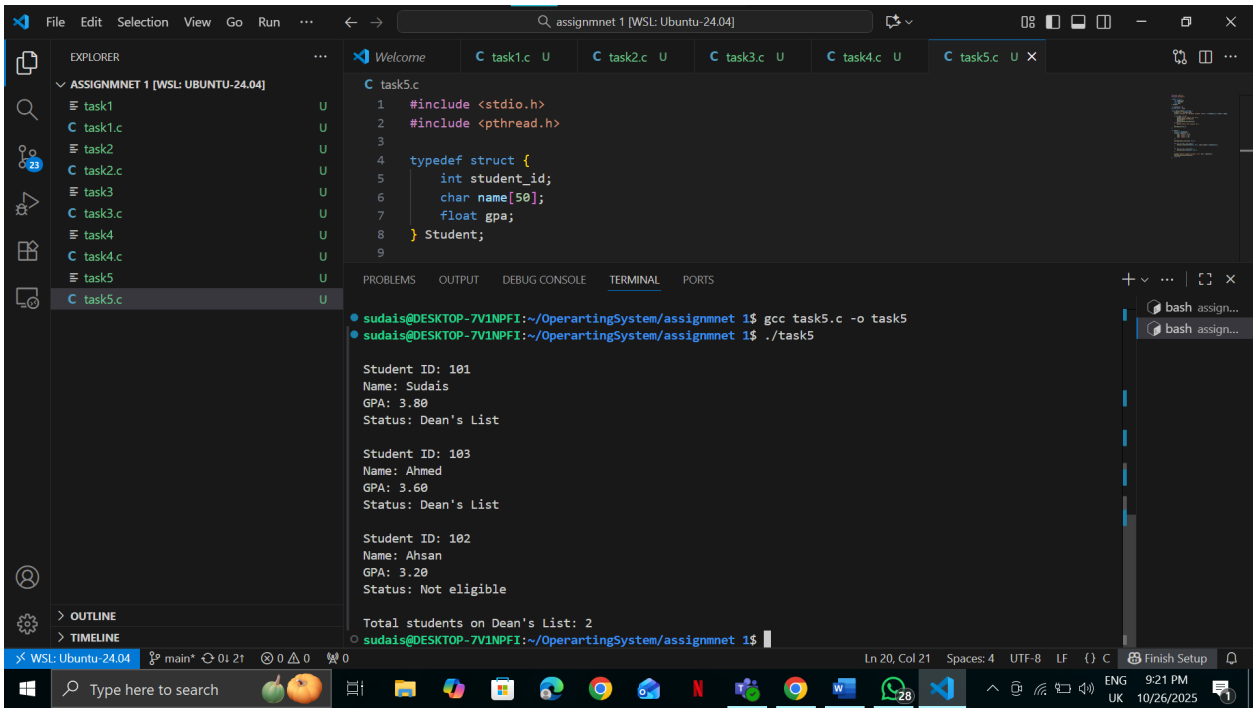
CODE:

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

```c
#include <pthread.h>

typedef struct {
    int student_id;
    char name[50];
    float gpa;
} Student;

int deanCount = 0;
pthread_mutex_t lock;

void* check_student(void* arg) {
    Student* s = (Student*)arg;
    printf("\nStudent ID: %d\nName: %s\nGPA: %.2f\n", s->student_id, s->name, s->gpa);

    if (s->gpa >= 3.5) {
        printf("Status: Dean's List \n");
        pthread_mutex_lock(&lock);
        deanCount++;
        pthread_mutex_unlock(&lock);
    } else {
        printf("Status: Not eligible \n");
    }
    pthread_exit(NULL);
}

int main() {
    pthread_t threads[3];
    Student students[3] = {
        {101, "Sudais", 3.8},
        {102, "Ahsan", 3.2},
        {103, "Ahmed", 3.6}
    };

    pthread_mutex_init(&lock, NULL);

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

    for (int i = 0; i < 3; i++) {
        pthread_join(threads[i], NULL);
    }
```

```
    printf("\nTotal students on Dean's List: %d\n", deanCount);
    pthread_mutex_destroy(&lock);
    return 0;
}
```

- **OUTPUT:**



Section-B: Short Questions

- **Answer all questions briefly and clearly.**

1. **Define an Operating System in a single line.**
   An Operating System (OS) is system software that manages computer hardware and software resources and provides services for programs.

2. **What is the primary function of the CPU scheduler?**
   The CPU scheduler decides which process from the ready queue should be executed next. It ensures efficient CPU utilization and fair process execution.

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

   There are three states of a process are:

   - Ready
   - Running
   - Waiting (or Blocked)

4. **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 (like process ID, state, registers, memory info).

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

Aspect	**Process**	**Program**
Definition	A program is a passive set of instructions stored on disk.	A process is an active instance of a program in execution.
State	It is static and does not perform any action.	It is dynamic and performs actions when executed.
Memory usage	Does not occupy CPU or memory until executed.	Occupies CPU time and memory while running.

6. **What do you understand by context switching?**
   Context switching is the process of saving the state of a running process and loading the state of another process to resume its execution

7. **Define CPU utilization and throughput.**

   - **CPU Utilization:** Percentage of time the CPU is actively working (not idle).
   - **Throughput:** Number of processes completed per unit time.

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

9. **How is waiting time calculated in process scheduling?**
   Waiting time is calculated by:

   Waiting time = Turnaround time – Burst (execution) time.

10. **Define response time in CPU scheduling.**
    Response time is the time from when a request is submitted until the first response is produced.

11. **What is preemptive scheduling?**
    Preemptive scheduling allows the CPU to be taken away from a running process before it finishes its execution.

12. **What is non-preemptive scheduling?**
    Non-preemptive scheduling means once a process starts executing, it cannot be interrupted until it finishes.

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

    Advantages of Round Robin:

    - Provides fair CPU time to all processes.
    - Suitable for time-sharing systems.

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

    Drawback of SJF:

    It may cause starvation for long processes.

15. **Define CPU idle time.**

CPU idle time is the period when the CPU is not executing any process.

16. **State two common goals of CPU scheduling algorithms.**

Goals of CPU scheduling:

- Maximize CPU utilization.
- Minimize waiting and response time

17. **List two possible reasons for process termination.**

Reasons for process termination:

- Process completed execution.
- Process killed due to error or user request

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

- wait(): Makes a parent process wait until its child process finishes.
- exit(): Used by a process to terminate its execution.

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

Aspect	**Shared memory**	**Message-passing**
Communication	Processes share a common memory space.	Processes exchange messages through the OS.
Speed	Faster (direct memory access).	Slower (involves system calls).

20. **Differentiate between a thread and a process.**

Aspect	**Thread**	**Process**

Definition	Smallest unit of CPU execution within a process.	Independent program in execution.
Memory	Shares memory with other threads of the same process.	Has its own memory space.
Overhead	Lower creation and management cost.	Higher creation and management cost.

21. **Define multithreading.**
    Multithreading is the ability of a CPU to execute multiple threads of a process concurrently.

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

- CPU-bound process: Spends most time using the CPU.
- I/O-bound process: Spends most time waiting for I/O operations.

23. **What are the main responsibilities of the dispatcher?**
    The **dispatcher** gives control of the CPU to the selected process, performs context switching, and starts its execution.

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

- Starvation: A process waits indefinitely due to lack of CPU allocation.
- Aging: Gradually increasing a process's priority to prevent starvation.

25. **What is a time quantum (or time slice)?**
    A time quantum (or time slice) is the fixed time a process is allowed to run before being preempted.

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

- Too large: System behaves like FCFS (poor response time).
- Too small: Too many context switches, increasing overhead.

27. **Define the turnaround ratio (TR/TS).**
Turnaround ratio (TR/TS) = Turnaround time / Service (burst) time. It shows how efficiently a process was handled.

28. **What is the purpose of a ready queue?**
A ready queue holds all processes that are ready to execute but waiting for CPU allocation.

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

Aspect	**CPU Burst**	**I/O Burst**
Definition	Time spent by process using the CPU.	Time spent waiting for I/O operations.
Activity	Execution of instructions.	Input/output operations (like disk or keyboard).

30. **Which scheduling algorithm is starvation-free, and why?**
Round Robin (RR) is starvation-free because each process gets an equal share of CPU time in a cyclic order.

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

Steps in UNIX process creation:

- The parent process calls `fork( )` to create a child.
- The child process gets a copy of the parent's data.
- The child calls `exec( )` to load a new program.
- The parent may use `wait( )` to wait for the child to finish.

32. **Define zombie and orphan processes.**

- Zombie process: A process that has finished execution but still has an entry in the process table (waiting for parent to read exit status).
- Orphan process: A process whose parent has terminated before it finishes.

33. **Differentiate between Priority Scheduling and Shortest Job First (SJF). 35. Define context switch time and explain why it is considered overhead.**

Aspect	Priority Scheduling	Shortest Job First
Basis	CPU assigned based on priority level.	CPU assigned to the process with the shortest burst time.
Issue	Can cause starvation of low-priority processes.	Can cause starvation of long jobs.

- **Context switch time** is the time taken by the CPU to save and load process states during a switch. It's considered **overhead** because no useful work is done during this time.

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

Three levels of schedulers:

- Long-term scheduler: Selects which jobs enter the ready queue.
- Short-term scheduler: Selects which ready process runs next on CPU.
- Medium-term scheduler: Suspends and resumes processes to balance load.

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

Aspect	User Mode	Kernel Mode
Access level	Limited access to system resources.	Full access to all hardware and memory.
Usage	Runs user applications.	Runs OS code and system calls.

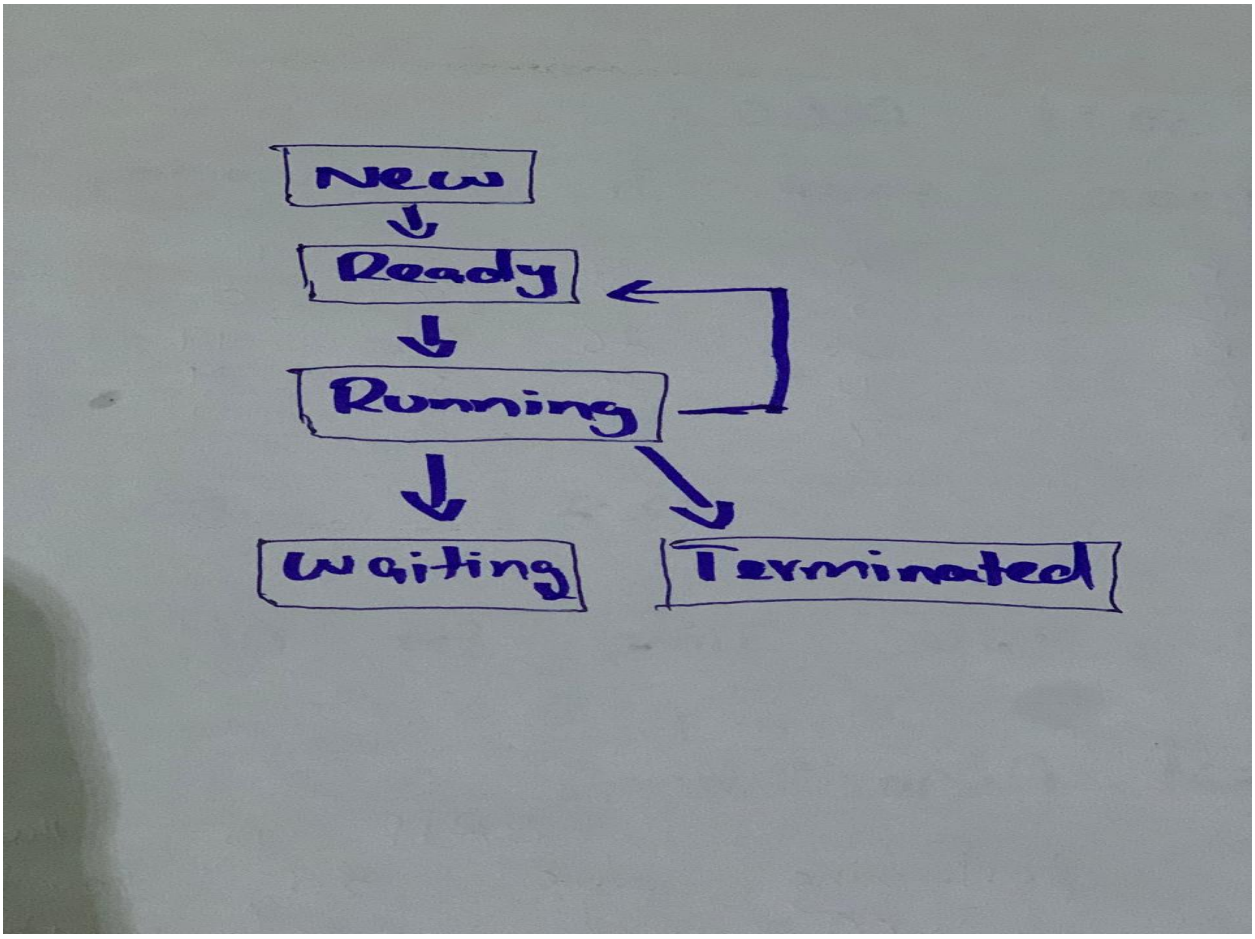
Section-C: Technical / Analytical Questions (4 marks each)

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

Process Life Cycle States:

1. **New** – Process is being created.
2. **Ready** – Process is waiting in main memory to be assigned to the CPU.
3. **Running** – Process is currently executing instructions on the CPU.
4. **Waiting (Blocked)** – Process is waiting for an I/O event or resource.
5. **Terminated** – Process has finished execution.

State Transitions Diagram:



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 one process or thread to another.
It involves saving the current process state and loading the next process state.

Overhead:

- Time taken to perform the switch (no useful work done during this time).
- Frequent context switches reduce CPU efficiency.

Information Saved/Restored:

- CPU registers and program counter (PC)
- Stack pointer and memory management info
- Process state (Ready, Running, etc.)
- I/O and open file information
- Accounting info (CPU time used)

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

A **PCB** is a data structure maintained by the operating system for each process.
It stores all information required to manage and resume a process.

Components:

1. **Process ID (PID)** – Unique identifier for the process.
2. **Process State** – New, Ready, Running, Waiting, Terminated.
3. **Program Counter (PC)** – Address of the next instruction to execute.
4. **CPU Registers** – Stored values during context switch.
5. **Memory Management Info** – Page tables, base/limit registers.
6. **Accounting Info** – CPU usage, priority, process time.
7. **I/O Status Info** – Open files, I/O devices allocated.

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

Scheduler	Function	Example	Frequency
Long-Term (Job Scheduler)	Selects which processes are loaded into main memory from job pool.	Batch system deciding which jobs to admit.	Low
Medium-Term	Swaps processes between main memory and secondary storage to control multiprogramming degree.	Swapping out idle processes to disk.	Moderate
Short-Term	Decides which ready process will execute	Round Robin or	Very High

Scheduler	Function	Example	Frequency
(CPU Scheduler)	next on CPU.	FCFS scheduling.	

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

Criterion	Definition	Optimization Goal
CPU Utilization	Percentage of time CPU is busy.	Maximize
Throughput	Number of processes completed per unit time.	Maximize
Turnaround Time	Time from process submission to completion.	Minimize
Waiting Time	Total time a process spends waiting in ready queue.	Minimize
Response Time	Time from submission until first response/output.	Minimize

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

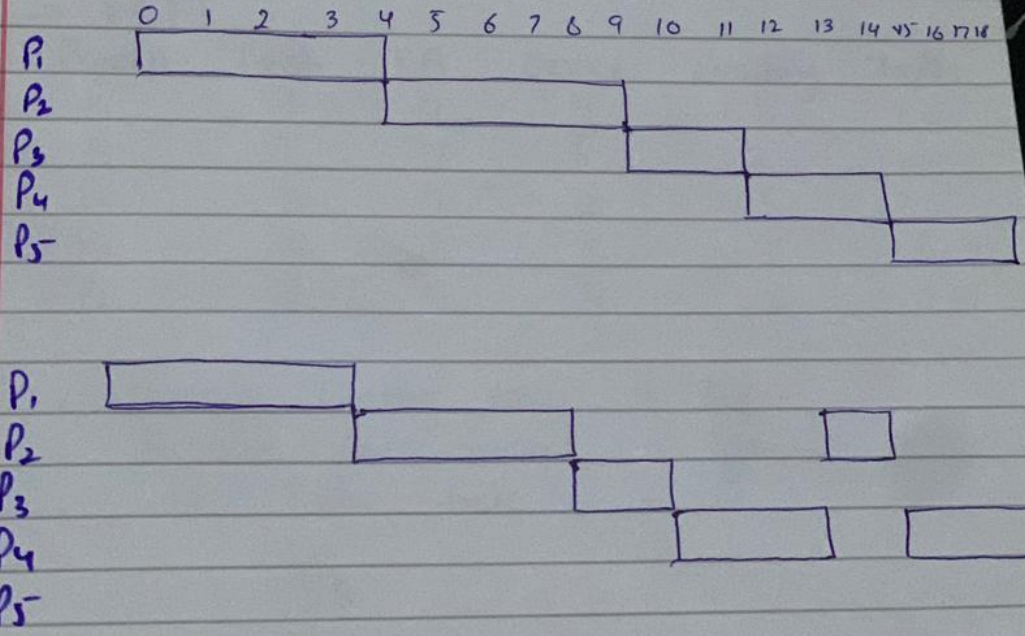
P5	9	4

SECTION - D

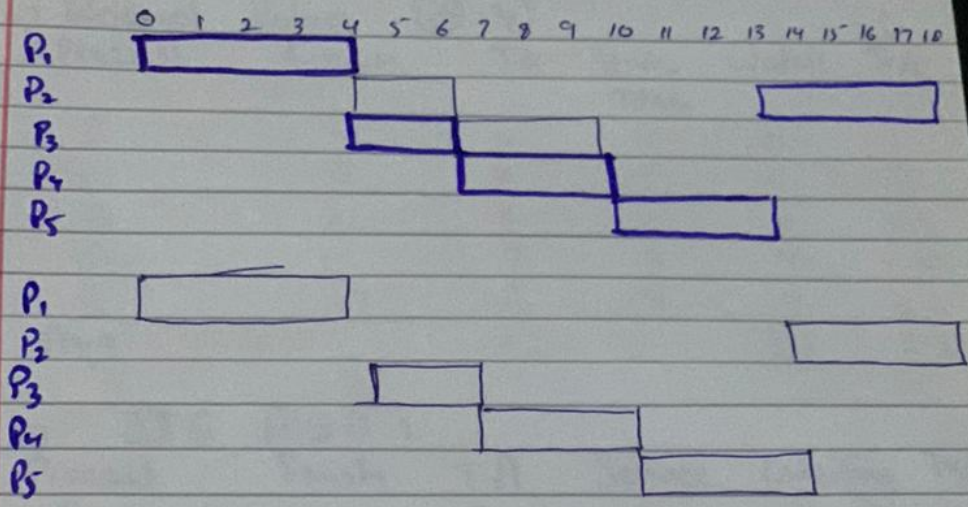
PART A :

Process	Arrival time	Service Time
P_1	0	4
P_2	2	5
P_3	4	2
P_4	6	3
P_5	9	4

a) Chant Chart



[5 + 0]



→ FCFS

Process	finish	T.A	Service	waiting	T.r/Ts
P_1	4	4	4	0	1.0
P_2	9	7	5	2	1.4
P_3	11	7	2	5	3.5
P_4	14	8	3	5	2.67
P_5	18	9	4	5	2.25

- Average waiting time = 3.4
- Average Turn around = 7.0
- CPU Idle time = 0

→ Round Robin (Q=4)

Process	Finish	T.A	Servic Time	Wating	Tr/Ts
P_1	4	4	4	0	1
P_2	18	16	5	11	3.2
P_3	10	6	2	4	3.0
P_4	13	7	3	4	2.37
P_5	17	8	4	4	2.0
Avg		8.2		4.6	2.32

→ SJF AlGO :

Process	Finish	T.A	Service	Waiting	Tr/Ts
P_1	4	4	4	0	1.0
P_2	18	16	5	11	3.2
P_3	6	2	2	0	1.0
P_4	9	3	3	0	1.0
P_5	13	4	4	0	1.0
Avg		5.8		2.2	1.44

→ SJRF ALGO :

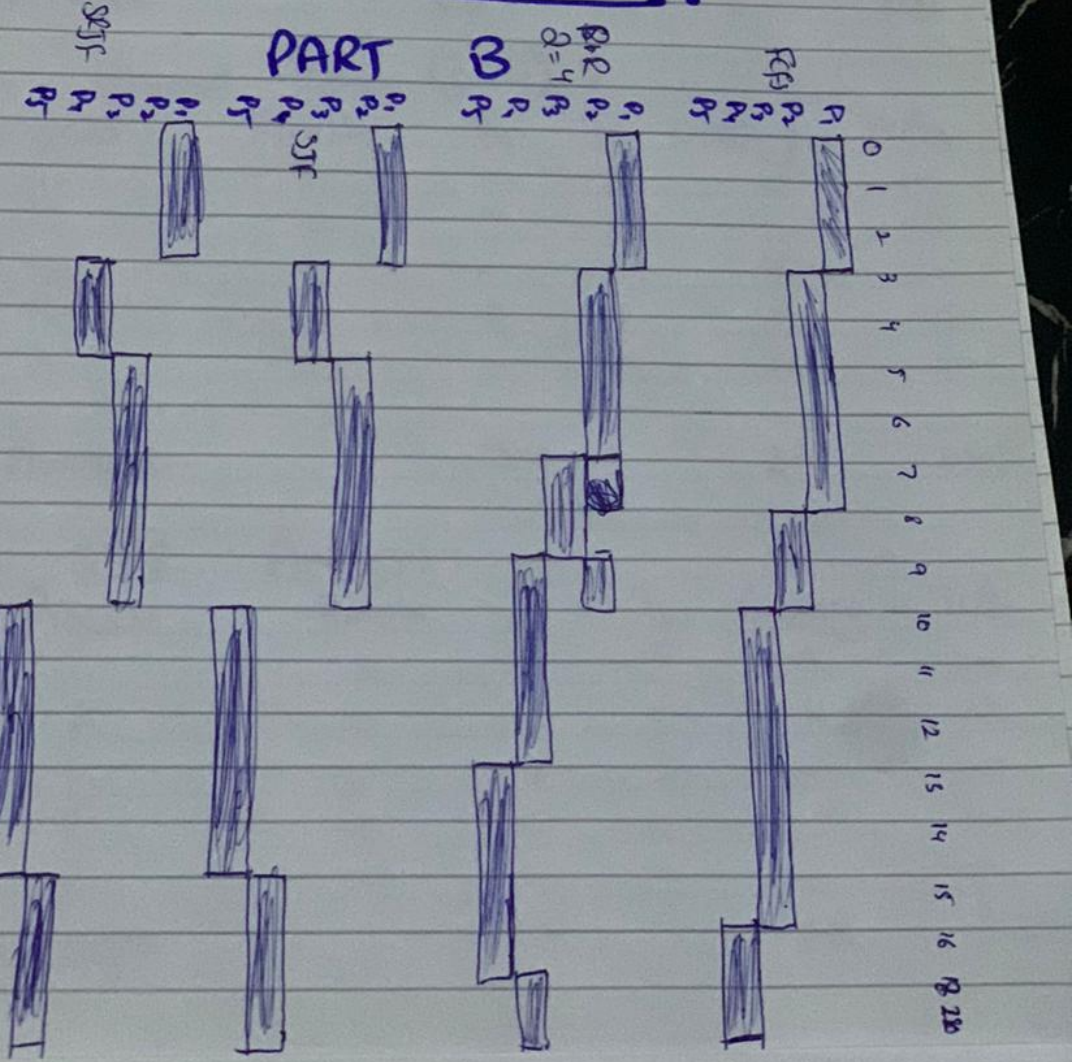
Process	Finish	Tr	Ts	Waiting	Tr/Ts
P_1	4	4	4	0	1.0
P_2	18	16	5	11	3.2
P_3	6	2	2	0	1.0
P_4	9	3	3	0	1.0
P_5	13	4	4	0	1.0
Average		5.8		2.2	1.44

CPU Idle Time for all = 0

⇒ Best Algorithm;

SJF and SRJF having the shortest waiting and turn around time. So, these are the most suitable.

PART B



Part-B

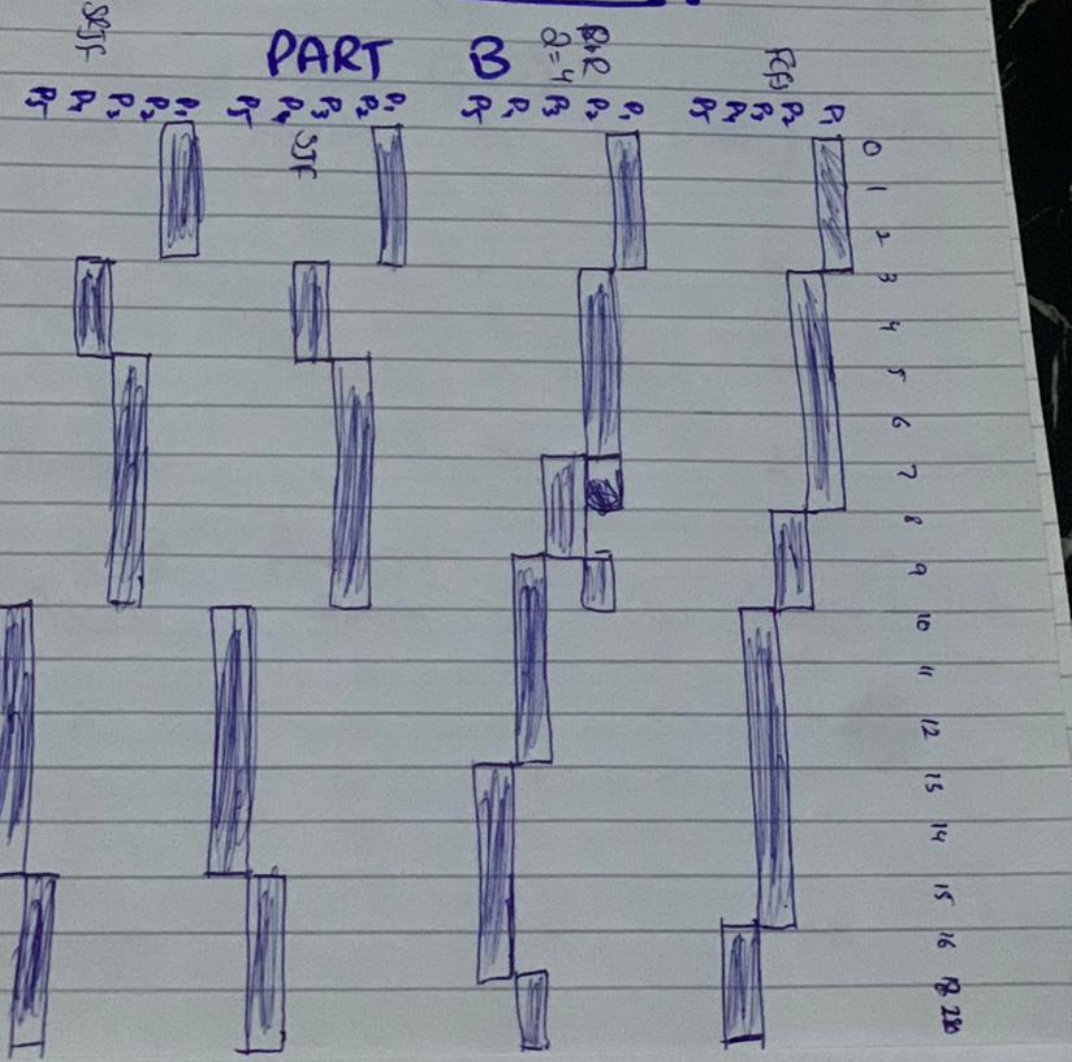
Process	Arrival Time	Service Time
P1	0	3
P2	1	5
P3	3	2
P4	9	6
P5	10	4

CPU Idle Time for all = 0

⇒ Best Algorithm;

SJF and SRJF having the shortest waiting and turn around time. So, these are the most suitable.

PART B



→ FCFS Analysis :

Process	Finish	Tr	Ts	Waiting	Tr/Ts
P_1	3	3	3	0	1.8
P_2	8	7	5	2	1.4
P_3	10	7	2	5	3.5
P_4	16	7	6	1	1.17
P_5	20	10	4	6	2.5
Avege		6.8		2.8	1.91

→ Round Robin ($\theta = 4$) :

Process	Finish	Tr	Ts	Waiting	Tr/Ts
P_1	3	3	3	0	1.0
P_2	16	9	5	4	1.8
P_3	9	6	2	4	3.0
P_4	20	11	6	5	1.83
P_5	18	8	4	4	2.0
Average		7.4		3.4	1.93

→ SJF Analysis :

Process	Finish	Tr	Ts	Waiting	Tr/Ts
P_1	3	3	3	0	1.0
P_2	10	9	5	4	1.8
P_3	5	2	2	0	1.0
P_4	20	11	6	5	1.83
P_5	14	14	4	0	1.0
Average		5.8		1.8	1.33

→ SRJF Analysis

Process	Finish	Tr	Ts	Waiting	Tr/Ts
P_1	3	3	3	0	1.0
P_2	10	9	5	4	1.8
P_3	5	2	2	0	1.0
P_4	20	11	6	5	1.8
P_5	14	14	4	0	1.0
Average		5.8		1.8	1.33

CPU Idle Time for All = 0

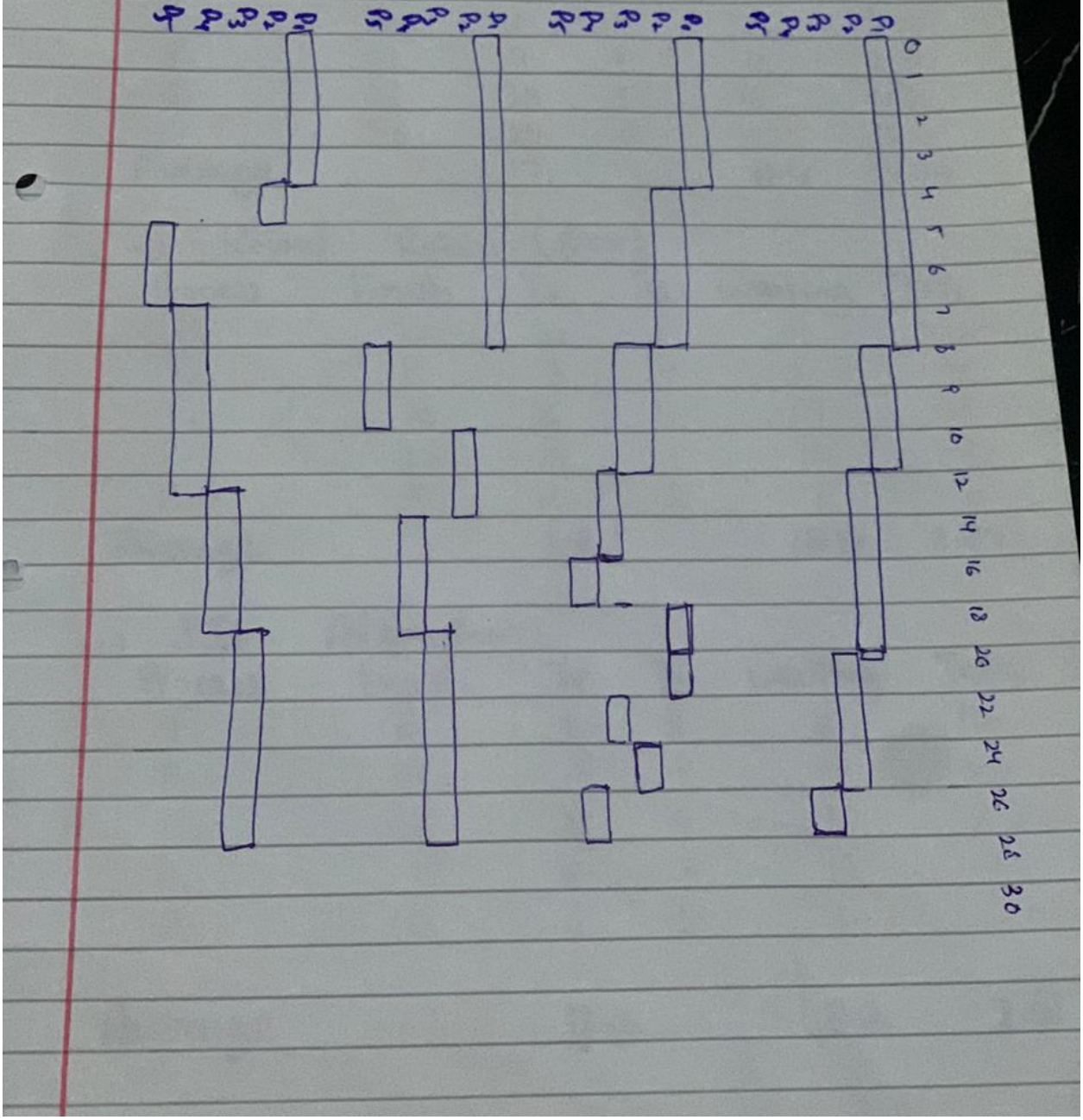
⇒ Best Algorithm:

SJF and SRJF are the best Performing for this data set. Tr/Ts is lowest as compare to the other. And Optimal for minimizing matrices

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

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

PART C



→ FCFS Analysis

Process	Finish	Tr	Ts	Waiting	Tr/Ts
P_1	8	8	8	0	1.0
P_2	12	11	4	7	2.75
P_3	21	19	9	10	2.11
P_4	26	23	5	18	4.60
P_5	28	24	2	22	12.0
Average		17		11.4	4.49

→ Round Robin (Q=4)

Process	finish	Tr	Ts	waiting	Tr/Ts
P_1	22	22	8	14	2.75
P_2	8	7	4	3	1.75
P_3	28	26	9	17	2.8
P_4	27	24	5	19	4.8
P_5	18	14	2	13	7.0
Average		18.6		13.6	3.84

→ SJF Algorithm:

Process	Finish	Tr	Ts	waiting	Tr/Ts
P_1	8	8	8	0	1.0
P_2	14	13	4	9	3.28
P_3	28	26	9	17	2.89
P_4	19	16	3	17	3.2
P_5	10	6	2	4	3.0
Average		13.8		8.2	2.67

→ SRJF ALGO :

Process	finish Tr	Ts	waiting Ti/Ts
P_1	19	8	11	2.38
P_2	5	4	4	0	1.0
P_3	28	26	9	17	2.89
P_4	12	9	5	4	1.8
P_5	7	3	2	1	1.5
Avg		12.2		66	1.91

CPU Idle Time for All = 0

⇒ Best Algorithm :

SRJF is the Best performing due to least Average Turn around and waiting time. for this data set.

- -

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.