National Textile University

Department of Computer Science

Subject:

Operating system

Submitted to:

Sir Naseer

Submitted by:

Ayema

Reg number:

23-NTU-CS-1142

Assignment no. :

01

Semester: 5Th

Section A

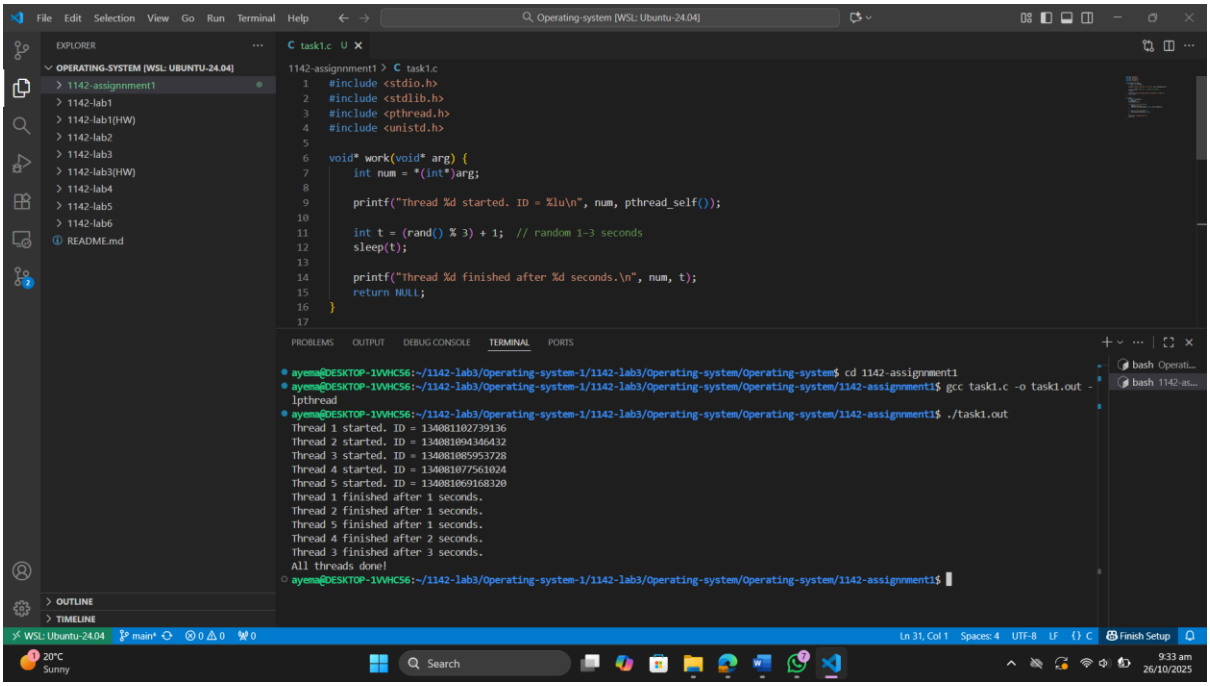
TASK 1 – Thread Information Display

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.

Code:

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

void* work(void* arg) {
    int num = *(int*)arg;

    printf("Thread %d started. ID = %lu\n", num, pthread_self());

    int t = (rand() % 3) + 1;  // random 1-3 seconds
    sleep(t);

    printf("Thread %d finished after %d seconds.\n", num, t);
    return NULL;
}

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

    for (int i = 0; i < 5; i++) {
        nums[i] = i + 1;
        pthread_create(&threads[i], NULL, work, &nums[i]);
    }

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

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

```



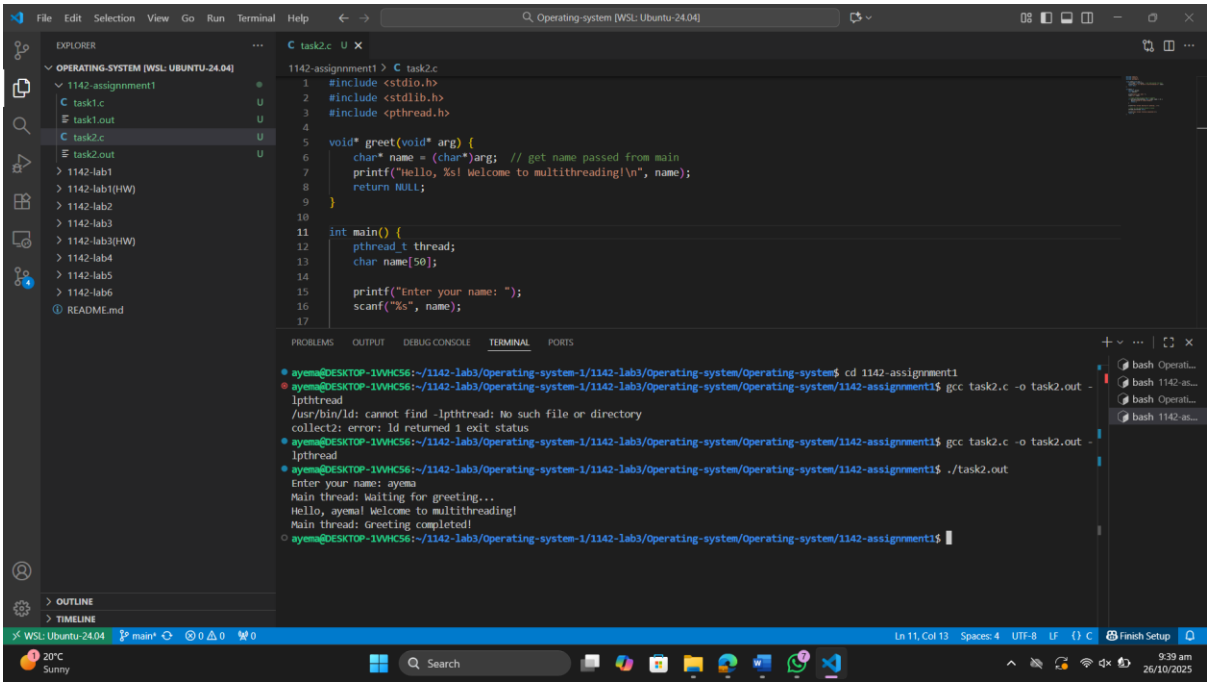
TASK 2– Personalized Greeting Thread5

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.

Code:

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

void* greet(void* arg) {
    char* name = (char*)arg;
    printf("Hello, %s! Welcome to multithreading!\n", name);
    return NULL;
}

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

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


    if (pthread_create(&thread, NULL, greet, name) != 0) {
        perror("Failed to create thread");
        return 1;
    }
    printf("Main thread: Waiting for greeting...\n");
    pthread_join(thread, NULL);

    printf("Main thread: Greeting completed!\n");
    return 0;
}

```

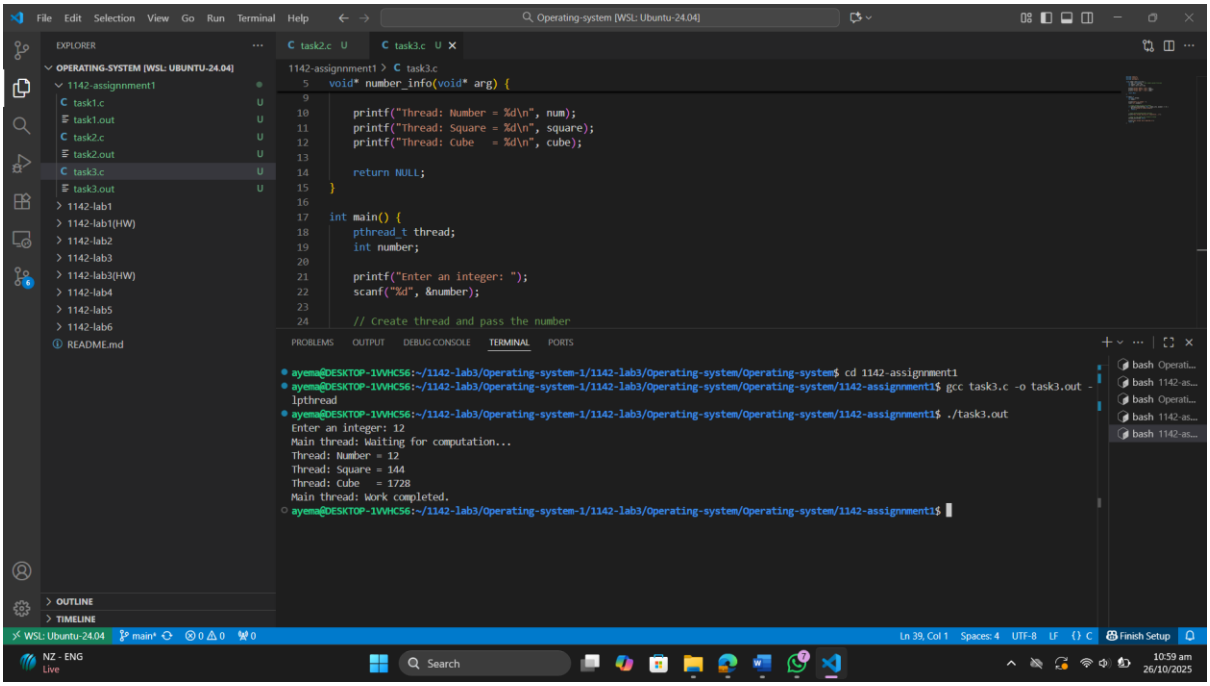


Task 3 – Number Info Thread

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.”

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

void* number_info(void* arg) {
    int num = *(int*)arg;
    int square = num * num;
    int cube = num * num * num;

    printf("Thread: Number = %d\n", num);
    printf("Thread: Square = %d\n", square);
    printf("Thread: Cube   = %d\n", cube);

    return NULL;
}

int main() {
    pthread_t thread;
    int number;

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

    if (pthread_create(&thread, NULL, number_info, &number) != 0) {
        perror("Failed to create thread");
        return 1;
    }

    printf("Main thread: Waiting for computation...\n");

    pthread_join(thread, NULL);

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

```

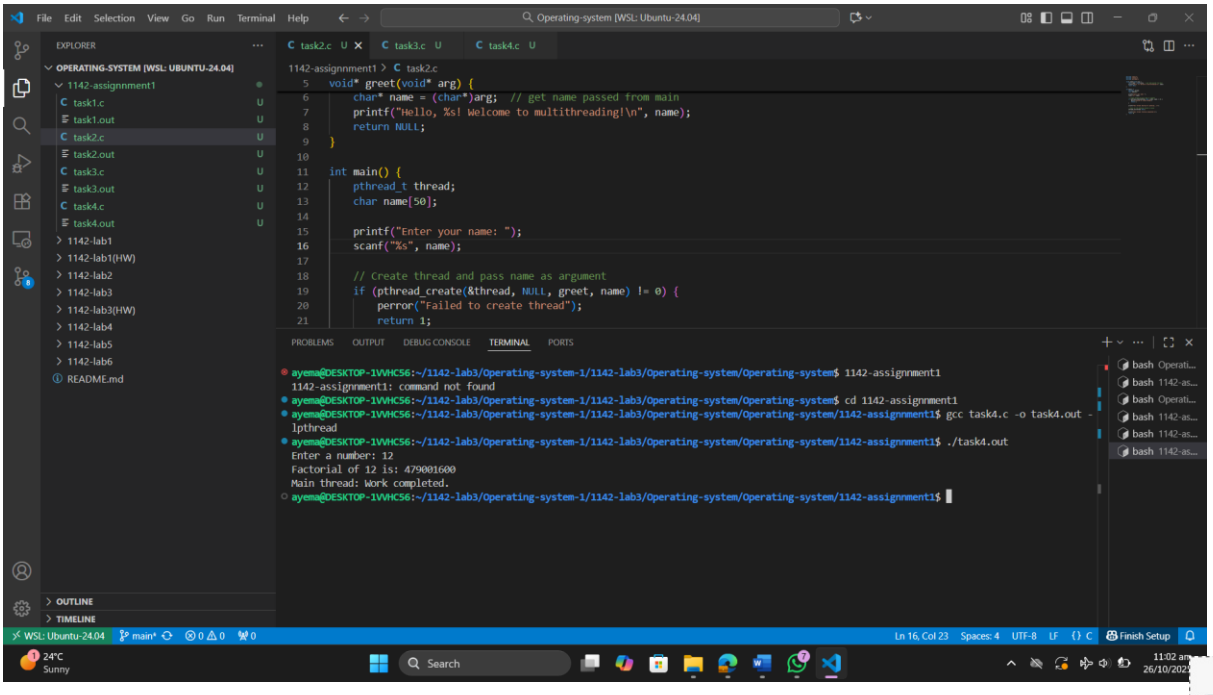


Task 4 – Thread Return Values

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.

```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 thread;
    int num;
    long long* fact;
    printf("Enter a number: ");
    scanf("%d", &num);

    if (pthread_create(&thread, NULL, factorial, &num) != 0) {
        perror("Failed to create thread");
        return 1;
    }
    pthread_join(thread, (void**)&fact);
    printf("Factorial of %d is: %lld\n", num, *fact);
    printf("Main thread: Work completed.\n");
    free(fact);
    return 0;
}

```

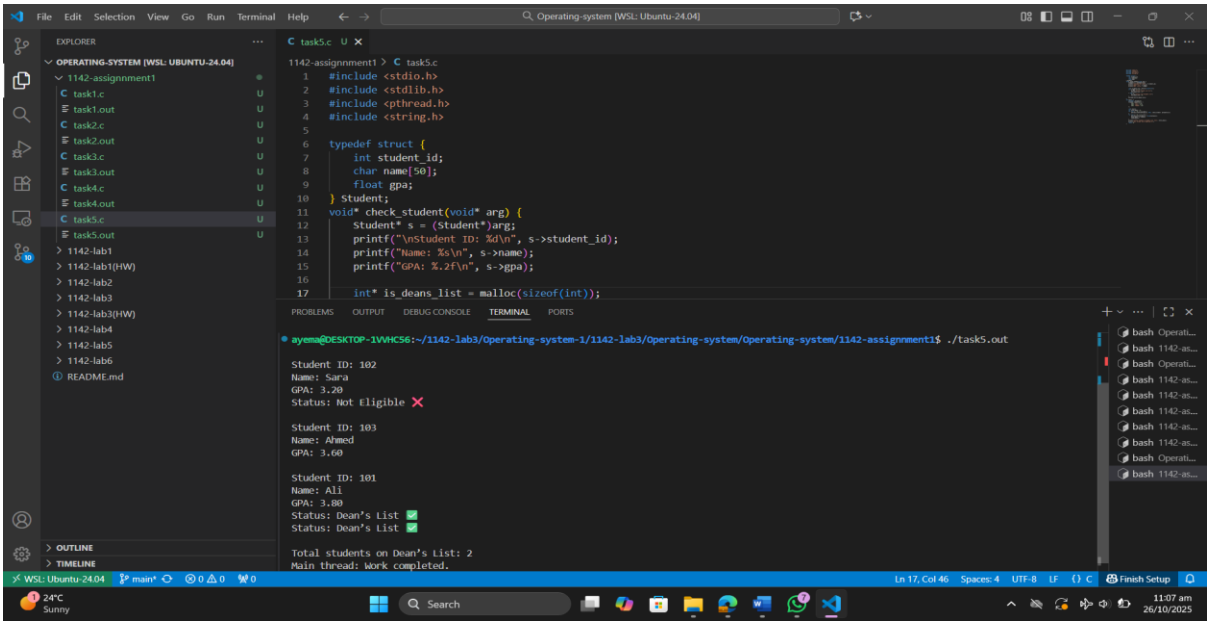


Task 5 – Struct-Based Thread Communication

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.

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

typedef struct {
    int student_id;
    char name[50];
    float gpa;
} Student;
void* check_student(void* arg) {
    Student* s = (Student*)arg;
    printf("\nStudent ID: %d\n", s->student_id);
    printf("Name: %s\n", s->name);
    printf("GPA: %.2f\n", s->gpa);

    int* is_deans_list = malloc(sizeof(int));
    if (s->gpa >= 3.5) {
        printf("Status: Dean's List ✅\n");
        *is_deans_list = 1;
    } else {
        printf("Status: Not Eligible ❌\n");
        *is_deans_list = 0;
    }
    pthread_exit(is_deans_list);
}
int main() {
    pthread_t threads[3];
    Student students[3] = {
        {101, "Ali", 3.8},
        {102, "Sara", 3.2},
        {103, "Ahmed", 3.6}
    };

    int* result;
    int total_deans = 0;
    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], (void**)&result);
        total_deans += *result;
        free(result);
    }
    printf("\nTotal students on Dean's List: %d\n", total_deans);
    printf("Main thread: Work completed.\n");
    return 0;
}

```



Section B

Answer all questions briefly and clearly.

1. **Define an Operating System in a single line?**

Operating system is an intermediary between User and hardware, it provides environment for program to execute.

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

The primary purpose of the CPU scheduler is to decide which process run next on CPU to optimize system performance and responsiveness.

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

Running- The process is prepared to run and waiting for CPU time.

Ready-The process is currently executed by CPU.

Blocked-The process is paused, waiting for I/O event.

4. **What is meant by a Process Control Block (PCB)?**
Data needed by OS to control the process. PCB is a data structure in OS containing important information about specific process such as process id, process state etc.

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

Process	Program
A program in execution.	A set of instructions stored on disk.
Active and dynamic.	Passive and static.
Needs CPU, memory, and resources.	Does not need resources until executed.
Example: Running MS Word.	Example: MS Word application file.

6. **What do you understand by context switching?**
   When CPU switches to another process, it must save the old state of process and load the saved state into another new process.

7. **Define CPU utilization and throughput.**
   - CPU utilization:
     It is the percentage of CPU is actively executing processes instead of idle.
     For example: If CPU is 90% of the time, CPU utilization = 90%
   - Throughput:
     It refers to the number of processes completed per unit of time.
     For example: If 10 processes finish in 1 second, so throughput=10

8. W**hat is the turnaround time of a process?**
   It is the total time a process needs to execute – the moment the process enters until the time of completion.
   Formula:
   Turnaround Time = Completion Time – Arrival Time5

9. **How is waiting time calculated in process scheduling?**
   It is the total time a process spends waiting in ready queue before it gets CPU before execution.
   Formula:
   Waiting Time=Turnaround Time−Burst Time

10. **Define response time in CPU scheduling.**
    It is the time from when a process is submitted until it first gets the CPU (i.e., starts execution for the first time).
    Formula:
    Response Time=First CPU Start Time−Arrival Time

11. **What is preemptive scheduling?**
    Currently running process may be interrupted and moved to ready state by the OS. Decision to preempt may be performed when a new process arrives, when an interrupt occurs that places a blocked process in the Ready state, or

periodically, based on a clock interrupt

12. What is non-preemptive scheduling?

Once a process is in the running state, it will continue until it terminates or blocks itself for I/O.

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

- **Fair:** Each process gets equal CPU time.
- **No starvation:** All processes get a chance to execute.

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

The Shortest Job First (SJF) algorithm may cause starvation because long processes might keep waiting if shorter ones keep arriving continuously.

15. Define CPU idle time.

CPU idle time is the time during which the CPU remains **idle or not executing any process** because no process is ready to run.

16. State two common goals of CPU scheduling algorithms.

- **Maximize CPU utilization** — keep the CPU as busy as possible.
- **Minimize waiting and turnaround time** — ensure faster and more efficient process execution.

17. List two possible reasons for process termination.

- The process completes its execution successfully.
- The process is killed or stopped due to an error or external request.

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

- The **wait ()** system call allows a parent process to pause its execution until one of its child processes finishes, so it can retrieve the child's exit status.
- The **exit ()** system call is used by a process to terminate its execution and return a status code to the operating system (or to the parent process).

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

Shared Memory	Message Passing
Processes share a common memory space.	Processes send and receive messages to communicate.
Very fast as data is directly accessed.	Slower due to message copying.

Needs synchronization (like semaphores).	No need for manual synchronization.
Suitable for large data transfer.	Suitable for small data exchange.
Example: Producer-Consumer model.	Example: Client-Server communication.

20. Differentiate between a thread and a process.

Thread	Process
A lightweight part of a process.	A heavy, independent program in execution.
Threads share memory and resources.	Each process has its own memory and resources.
Faster to create and switch.	Slower to create and switch.
Failure of one thread may affect others.	One process failure doesn't affect others.
Used in multitasking within the same app.	Used for running separate applications.

21. Define multithreading.

Multithreading is the ability of a CPU or a program to run multiple threads concurrently within a single process to improve performance and responsiveness.

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

CPU-Bound Process	**I/O-Bound Process**
Spends most of the time using the CPU.	Spends most of the time waiting for I/O operations.
Performs heavy calculations.	Performs frequent input/output tasks.
Example: Video rendering or data processing.	Example: Reading files or printing documents.
Needs a fast processor.	Needs fast I/O devices.

23.**What are the main responsibilities of the dispatcher?**

- Gives control of the CPU to the process selected by the scheduler.

- Performs context switching between processes.
- Switches to user mode.
- Jumps to the correct program location to restart the process.
-

24.Define starvation and aging in process scheduling.

- Starvation: It is a situation in which a runnable process is overlooked indefinitely by the scheduler, although it can process, that is never chosen.
- Aging: Gradually increasing a process's priority to prevent starvation.

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

It is the fixed time period for which 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).

It is the ratio of Turnaround Time (TR) to Service Time (TS).
It measures how efficiently a process is completed compared to its actual service time.

turnaround ratio = TR/TS

28.What is the purpose of a ready queue?

The purpose of ready queue is that t holds all the processes that are ready to execute and waiting for CPU time. The scheduler selects processes from this queue to execute next.

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

CPU Burst	I/O Burst
A period when a process is executing instructions on the CPU.	A period when a process is waiting for input/output operations to complete.
Involves calculations and logic operations.	Involves waiting for devices like disk, printer, or keyboard.
Example: Performing arithmetic operations.	Example: Reading data from a file.

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

Round Robin (RR) is starvation-free because every process gets CPU time in a circular order for a fixed time slice, ensuring no process waits indefinitely.

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

- Parent creates a child process using fork ().
- OS allocates memory and resources.
- Child loads a new program with exec ().
- Process starts running.
- Parent uses wait () when the child ends.

32.Define zombie and orphan processes.

- Zombie Process: Occurs when a process completes execution, but its parent has not yet read its exit status using wait (), leaving an entry in the process table.
- Orphan Process: Occurs when the parent process terminates before its child, leaving the child process under the care of the init process.

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

Priority Scheduling	Shortest Job First (SJF)
Processes are executed based on priority levels.	Processes with the shortest CPU burst are executed first.
High-priority processes may cause starvation of low-priority ones.	Shorter processes may cause longer ones to starve.
Priorities can be static or dynamic.	Based on estimated or known CPU time.

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

It is the time taken by the CPU to save the state of one process and load the state of another. This process ensures multitasking but is considered overhead since it consumes CPU time without doing useful work.

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

- Long-term scheduling: The decision to add to the pool of processes to be executed.
- Medium-term scheduling: The decision to add to the number of processes that are partially or fully in main memory
- Short-term scheduling: The decision as to which available process will be executed by the processor.

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

User Mode	Kernel Mode
Executes user applications with limited privileges.	Executes OS code with full system privileges.
Cannot directly access hardware or memory.	Can access all hardware and memory directly.
Errors affect only the user program.	Errors can affect the entire system.
Example: Running a web browser.	Example: Handling system calls or device drivers.

Section C

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:

A process goes through several states during its lifetime — from creation to termination.

The main states are: New, Ready, Running, Waiting, and Terminated.

1. New:

The process is being created and initialized by the operating system.

2. Ready:

The process is loaded into main memory and waiting for CPU time.

3. Running:

The process is currently being executed by the CPU.

4. Waiting (Blocked):

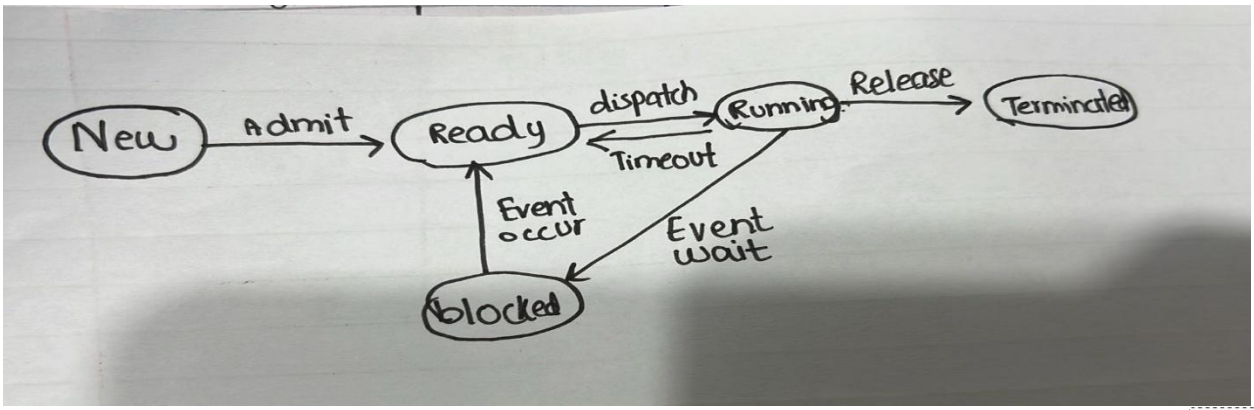
The process is waiting for some event like I/O completion or resource availability.

5. Terminated:

The process has finished execution and is removed from the process table.

State Transitions:

- New → Ready: Process is admitted by the scheduler.
- Ready → Running: Process is assigned to the CPU.
- Running → Waiting: Process requests I/O or an event.
- Waiting → Ready: I/O or event completes.
- Running → Terminated: Process completes or is killed.
- Running → Ready: CPU preemption (time slice over).



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

A context switch happens when the CPU changes from one process (or thread) to another.

The time spent saving and loading process information is called context switch overhead.

It is considered overhead because it uses CPU time but does not perform actual processing work.

Information Saved and Restored:

When a context switch occurs, the operating system saves and later restores:

1. Program Counter – Next instruction address.
2. CPU Registers – Current working data.
3. Process State – Whether the process is running, waiting, or ready.
4. Memory Information – Details like base and limit registers.
5. I/O and Accounting Info – Open files, CPU time used, and process priority.

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 files

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

Feature	Long-Term Scheduler	Medium-Term Scheduler	Short-Term Scheduler

Work	Decides which new processes enter memory.	Suspends or resumes processes.	Chooses which process runs next on CPU.
Speed	Slow	Medium	Fast
When used	When a new job arrives.	When system is overloaded.	Very often during execution.
Goal	Control number of running processes.	Manage memory use.	Give fair CPU time to all.
Example	Loading new program from disk.	Swapping a process to free memory.	Selecting next process in Round Robin.

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

1. CPU Utilization:
    - Meaning: Measures how effectively the CPU is being used.
    - Goal: Keep the CPU as busy as possible (maximize utilization).
2. Throughput:
    - Meaning: Number of processes completed per unit time.
    - Goal: Increase throughput to complete more processes in less time.
3. Turnaround Time:
    - Meaning: Total time taken from process submission to completion.
    - Goal: Minimize turnaround time to make processes finish faster.
4. Waiting Time:
    - Meaning: Total time a process spends waiting in the ready queue.
    - Goal: Reduce waiting time to improve overall efficiency.
5. Response Time:

- Meaning: Time from process submission until the first response is produced.
- Goal: Minimize response time for better interactivity (important in real-time systems).

Section D

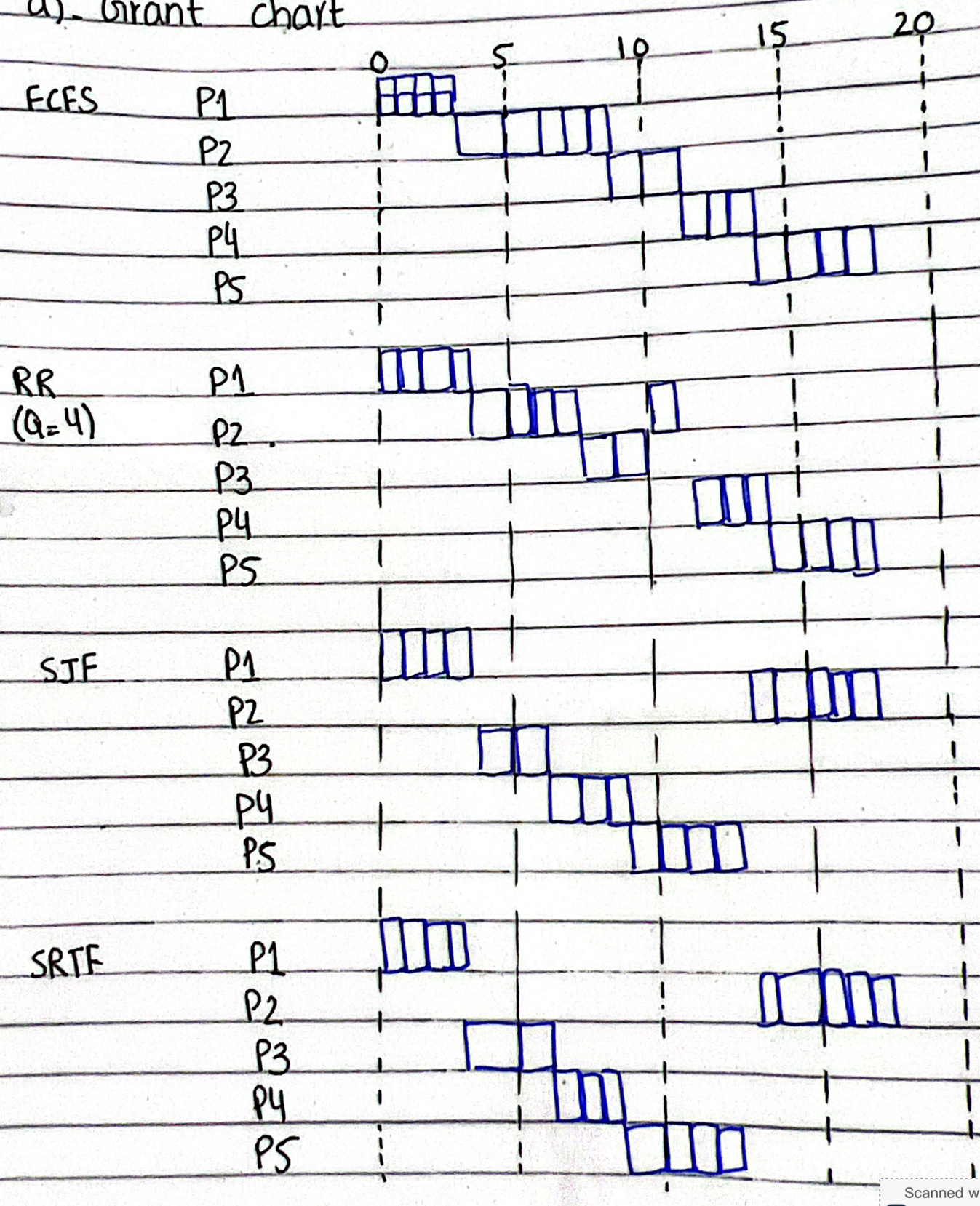
pages are attached below for this section.

Section-D

Part A:

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

a). Grant chart



b)

FCFS:

Process	Waiting Time	Turnaround time	TR/TS ratio	CPU Idle
P1	0	4	1.00	0
P2	2	7	1.40	0
P3	5	7	3.50	0
P4	5	8	2.67	0
P5	5	9	2.25	0
Average:	3.4	7.0	2.16	0

RR : (Q=4)

Process	waiting time	Turnaround Time	TR/TS ratio	CPU Idle
P1	0	4	1.00	0
P2	4	9	1.80	0
P3	4	6	3.00	0
P4	5	8	2.67	0
P5	5	9	2.25	0
Average	3.6	7.2	2.14	

SJF :

Process	waiting Time	Turnaround time	TR/TS ratio	CPU Idle
P1	0	4	1.00	0
P2	11	16	3.20	0
P3	0	2	1.00	0
P4	0	3	1.00	0
P5	0	4	1.00	0
Average	2.2	5.8	1.44	

SRTF :

Process	waiting time	Turnaround time	TR/TS	CPU Idle
P1	0	4	1.00	0
P2	11	16	3.20	0
P3	0	2	1.00	0
P4	0	3	1.00	0
P5	0	4	1.00	0
Average	2.2	5.8	1.44	

c) Compare average value and identify which algorithm perform best:

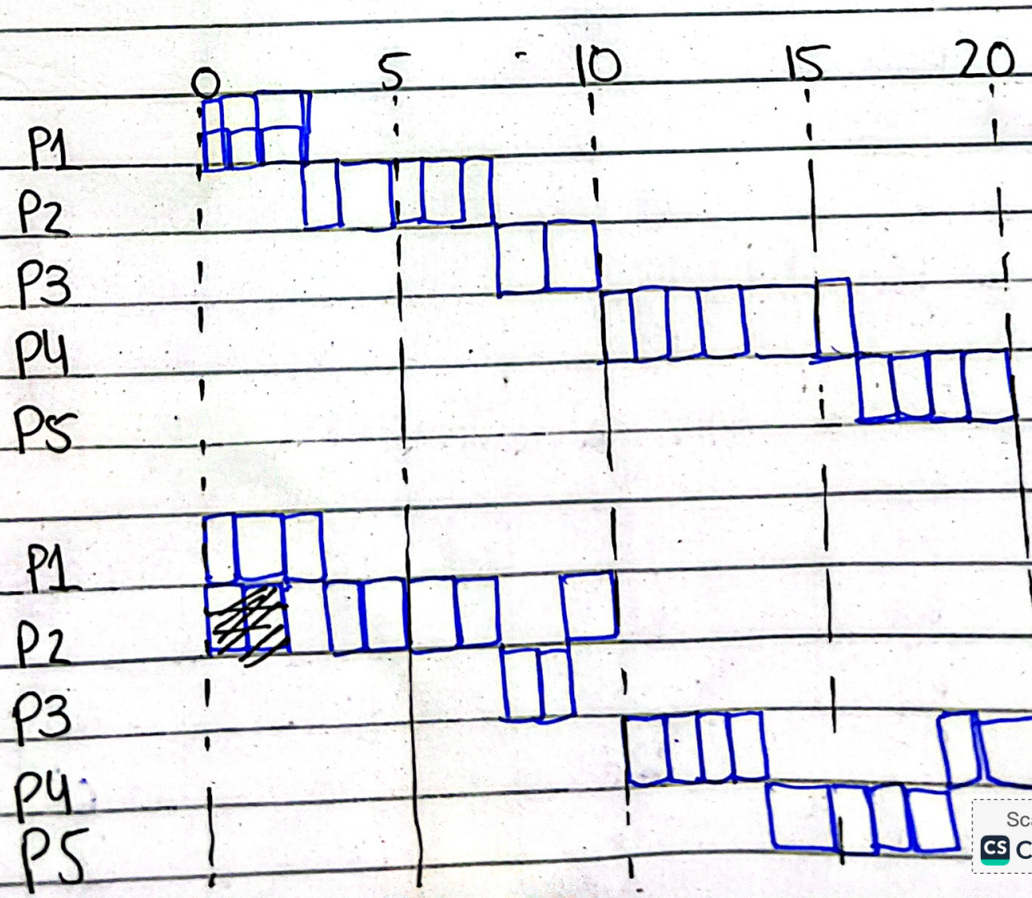
Algorithm	Avg waiting Time	Average Turnaround Time	Avg TR/RS ratio
FCFS	3.4	7.0	2.16
RR(q=4)	3.6	7.2	2.14
SJF	2.2	5.8	1.44
SRTF	2.2	5.8	1.44

SJF and SRTF algorithm perfom best because they have good (low) avg waiting time, avg. Turnaround time and avg. TR/RS. ratio..

Part B:

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

a) Grant graph



b)

FCFS:

Process	waiting time	Turnaround Time	TR/TS	CPU Idle
P1	0	3	1.00	0
P2	2	7	1.40	0
P3	5	7	3.50	0
P4	3	9	1.50	0
P5	8	12	3.00	0
Average	3.6	7.6	2.08	

RR (q=4) :

Process	waiting time	Turnaround Time	TR/TS	CPU Idle
P1	0	3	1.00	0
P2	9	14	2.80	0
P3	4	6	3.00	0
P4	14	20	3.33	0
P5	11	15	3.75	0
Average	7.6	11.6	2.78	

SJF:

Process	waiting time	Turnaround time	TR/TS	CPU Idle
P1	0	3	1.00	a
P2	4	9	1.80	0
P3	0	2	1.00	0
P4	9	15	2.50	0
P5	4	8	2.00	0
Average	3.4	7.4	1.66	

SRTF:

Process	waiting time	Turnaround Time	TR/TS	CPU Idle
P1	0	3	1.00	0
P2	4	9	1.80	0
P3	0	2	1.00	0
P4	9	15	2.50	0
P5	4	8	2.00	0
Average	3.4	7.4	1.66	

c). Compare average value and identify which algorithm is best.

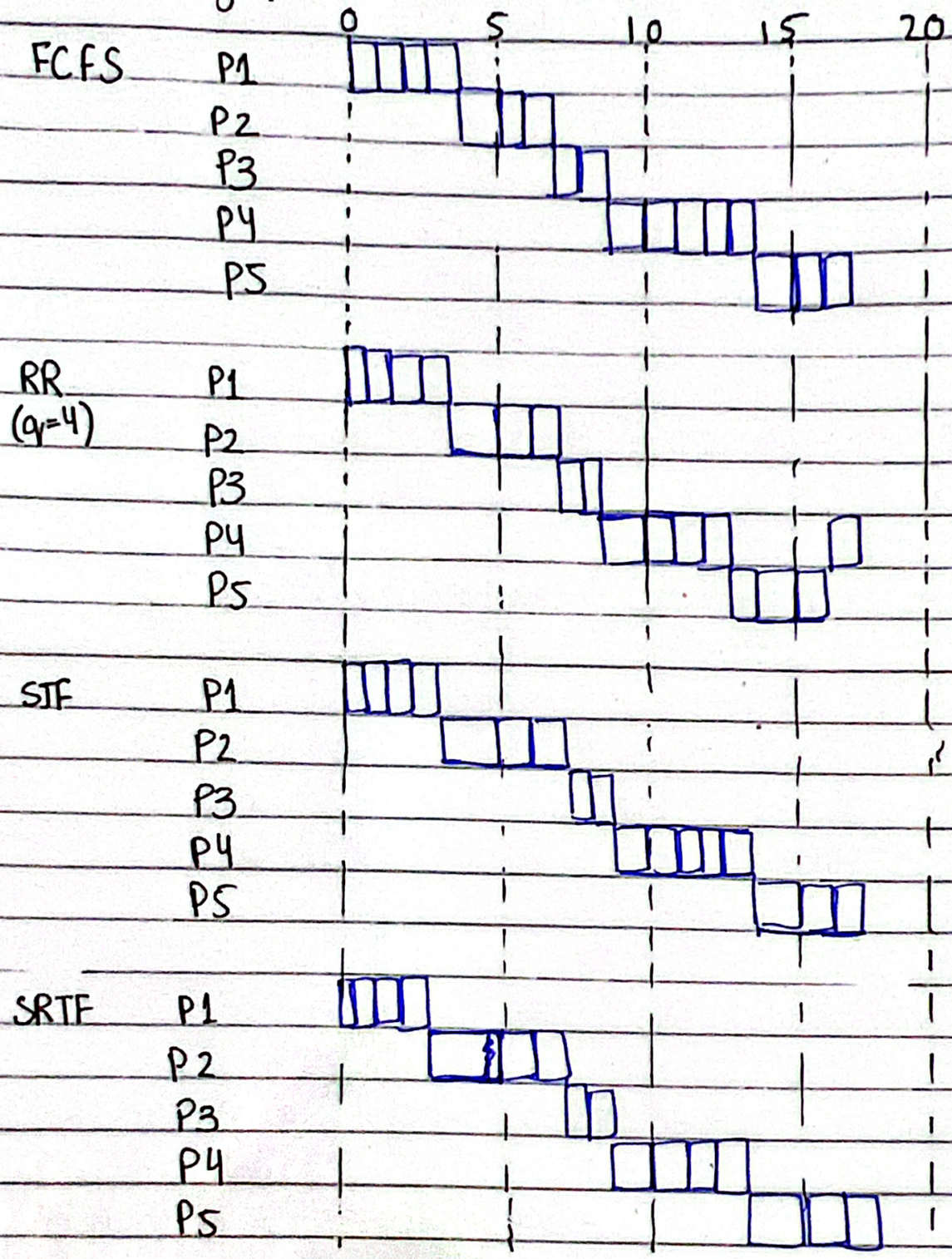
Algorithm	Avg. waiting time	Avg. Turnaround time	Averge TR/TS ratio
FCFS	3.6	7.6	2.08
RR(q=4)	7.6	11.6	2.78
SJF	3.4	7.4	1.66
SRTF	3.4	7.4	1.66

According to above table, SJF and SRTF is the best algorithm to choose

Part C: your own arrival time and service time:

Process	Arrival time	Service time
P1	0	4
P2	3	3
P3	6	2
P4	9	5
P5	12	4

a)- Grant graph



b)-

FCFS:

Process	waiting time	Turnaround time	TR/TS	CPU Idle
P1	0	4	1.00	0
P2	1	4	1.33	0
P3	2	4	2.00	0
P4	3	8	1.60	0
P5	6	10	2.50	0
Average	2.4	6.0	1.69	

RR: (q=4)

Process	Waiting time	Turnaround time	TR/TS	CPU Idle
P1	0	4	1.00	0
P2	1	4	1.33	0
P3	2	4	2.00	0
P4	7	12	2.40	0
P5	4	8	2.00	0
Average	2.8	6.4	1.75	

SJF:

Process	Waiting time	Turnaround time	TR/TS	CPU Idle
P1	0	4	1.00	0
P2	1	4	1.33	0
P3	2	4	2.00	0
P4	3	8	1.60	0
P5	6	10	2.50	0
Average	2.4	6.0	1.69	

SRTF:

Process	Waiting time	Turnaround time	TR/TS	CPU Idle
P1	0	4	1.00	0
P2	1	4	1.33	0
P3	2	4	2.00	0
P4	3	8	1.60	0
P5	6	10	2.50	0
Average	2.4	6.0	1.69	

Q) Compare algorithm value and identify which algorithm is best

Algorithm	Avg. waiting time	Avg. Turnaround time	Avg TR/TS
FCFS	2.4	6.0	1.69
RR (q=4)	2.8	6.4	1.75
SJF	2.4	6.0	1.69
SRTF	2.4	6.0	1.69

As, ~~FCFS = RR~~ (q=4) Avg. waiting time, avg turnaround time and avg TR/TS of FCFS, SJF and SRTF are same and lowest, so they are best algorithm to