Nagar Yuwak Shikshan Sanstha's
Yeshwantrao Chavan College of Engineering
(An Empowered Autonomous Institution affiliated to Rashtrasant Tukadoji Maharaj Nagpur University)
Hingna Road, Wanadongri, Nagpur - 441 110
NAAC A++
Ph.: 07104-237919, 234623, 329249, 329250 Fax: 07104-232376, Website: www.ycce.edu

Department of Computer Science & Engineering (IOT)

Vision of the Department

To be a well-known centre for pursuing computer education through innovative pedagogy, value-based education and industry collaboration.

Mission of the Department

To establish learning ambience for ushering in computer engineering professionals in core and multidisciplinary area by developing Problem-solving skills through emerging technologies.

Session 2025-2026

Vision: Dream of where you want.	**Mission:** Means to achieve Vision

Program Educational Objectives of the program (PEO): (broad statements that describe the professional and career accomplishments)

PEO1	**Preparation**	**P: Preparation**	**Pep-CL abbreviation pronounce as Pep-si-lL easy to recall**
PEO2	**Core Competence**	**E: Environment (Learning Environment)**	
PEO3	**Breadth**	**P: Professionalism**	
PEO4	**Professionalism**	**C: Core Competence**	
PEO5	**Learning Environment**	**L: Breadth (Learning in diverse areas)**	

Program Outcomes (PO): (statements that describe what a student should be able to do and know by the end of a program)

Keywords of POs:
Engineering knowledge, Problem analysis, Design/development of solutions, Conduct Investigations of Complex Problems, Engineering Tool Usage, The Engineer and The World, Ethics, Individual and Collaborative Team work, Communication, Project Management and Finance, Life-Long Learning

PSO Keywords: Cutting edge technologies, Research

"I am an engineer, and I know how to apply engineering knowledge to investigate, analyse and design solutions to complex problems using tools for entire world following all ethics in a collaborative way with proper management skills throughout my life." *to contribute to the development of cutting-edge technologies and Research*.

Integrity: I will adhere to the Laboratory Code of Conduct and ethics in its entirety.

Name and Signature of Student and Date
Bhushan V. Tayade
29-10-2025

Nagar Yuwak Shikshan Sanstha's
Yeshwantrao Chavan College of Engineering
(An Empowered Autonomous Institution affiliated to Rashtrasant Tukadoji Maharaj Nagpur University)
Hingna Road, Wanadongri, Nagpur - 441 110
NAAC A++
Ph.: 07104-237919, 234623, 329249, 329250 Fax: 07104-232376, Website: www.ycce.edu

Department of Computer Science & Engineering (IOT)

Vision of the Department

To be a well-known centre for pursuing computer education through innovative pedagogy, value-based education and industry collaboration.

Mission of the Department

To establish learning ambience for ushering in computer engineering professionals in core and multidisciplinary area by developing Problem-solving skills through emerging technologies.

Session	**2025-26 (ODD)**	**Course Name**	**Operating System Lab**
Semester	**5**	**Course Code**	**23IOT1504**
Roll Number	**035**	**Name of Student**	**Bhushan V. Tayade**

Practical Number	4
Course Outcome	1. Understand Computer System Configuration and Simulate system resources efficiently using Linux Commands (CO1) 2. Analyse operating system functionalities utilizing system calls, thread programming and process scheduling algorithms (CO2) 3. Apply Synchronization primitives to implement a Deadlock-free solution(CO3) 4. Simulate Disk scheduling, Memory allocation, File allocation, page replacement algorithms (CO4)
Aim	Write a simulator program for CPU Scheduling Algorithms. A. Non-Preemptive CPU Scheduling Algorithm (Any one) B. Preemptive CPU Scheduling Algorithm
Problem Definition	CPU Scheduling is a key function of the operating system that decides the order in which processes are executed by the CPU. The goal is to optimize CPU utilization, minimize waiting time, turnaround time, and response time, and ensure fairness among processes. This experiment focuses on implementing and simulating both Non-Preemptive (e.g., FCFS or SJF) and Preemptive (e.g., SRTF or Round Robin) CPU Scheduling algorithms. The simulation should accept process details such as process ID, arrival time, and burst time, then compute and display the Gantt chart, waiting time, turnaround time, and average metrics for both scheduling techniques.
Theory (100 words)	CPU Scheduling is the method by which processes are assigned to the CPU for execution. The scheduler decides which process should run at any given time based on specific algorithms. In Non-Preemptive Scheduling, once a process starts execution, it runs until completion (e.g., FCFS, SJF). In Preemptive Scheduling, a process can be interrupted and replaced by another process with a higher priority or shorter burst time (e.g., SRTF, Round Robin).

	Efficient scheduling reduces waiting and turnaround times while maximizing CPU utilization and throughput. The performance of these algorithms can be analyzed through simulation and comparison.
Procedure and Execution (100 Words)	Step for Implementation: 1. Start the program and include required header files: stdio.h, stdlib.h, and string.h. 2. Define a structure to store process details: Process ID, Arrival Time, Burst Time, Waiting Time, Turnaround Time, Completion Time, and Remaining Time. 3. Accept user input for the number of processes and their details (PID, Arrival Time, Burst Time). 4. For Non-Preemptive Algorithm (SJF or FCFS): • Sort processes based on arrival or burst time. • Calculate start, waiting, turnaround, and completion times sequentially. 5. For Preemptive Algorithm (SRTF or Round Robin): • Use a loop to simulate each time unit. • At each step, select the process with the shortest remaining time or the next in the ready queue. • Update remaining time, and when it reaches zero, compute completion metrics. 6. Display a Gantt chart showing the execution order of processes. 7. Compute and print results: Waiting time, Turnaround time, Average Waiting Time, and Average Turnaround Time. 8. End the program after displaying final scheduling results.

	Code:
	
	#include <stdio.h>
	
	struct process {
	int pid, arrival, burst, completion, waiting, turnaround, remaining;
	};

```
int main() {
    struct process p[10];
    int n, time = 0, completed = 0, min_index;
    float avg_wait = 0, avg_turn = 0;

    printf("Enter number of processes: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        p[i].pid = i + 1;
        printf("Enter Arrival Time and Burst Time for P%d: ", i + 1);
        scanf("%d %d", &p[i].arrival, &p[i].burst);
        p[i].remaining = p[i].burst;
    }

    printf("\n--- SRTF (Preemptive) CPU Scheduling ---\n");

    while (completed != n) {
        min_index = -1;
        int min_time = 9999;

        for (int i = 0; i < n; i++) {
            if (p[i].arrival <= time && p[i].remaining > 0 && p[i].remaining <
min_time) {
                min_time = p[i].remaining;
                min_index = i;
            }
        }

        if (min_index == -1) {
            time++;
            continue;
        }

        p[min_index].remaining--;
        time++;

        if (p[min_index].remaining == 0) {
            completed++;
            p[min_index].completion = time;
            p[min_index].turnaround = p[min_index].completion -
p[min_index].arrival;
```

```
            p[min_index].waiting = p[min_index].turnaround -
p[min_index].burst;
            avg_wait += p[min_index].waiting;
            avg_turn += p[min_index].turnaround;
        }
    }

    printf("\nPID\tAT\tBT\tCT\tTAT\tWT\n");
    for (int i = 0; i < n; i++) {
        printf("P%d\t%d\t%d\t%d\t%d\t%d\n",
            p[i].pid, p[i].arrival, p[i].burst,
            p[i].completion, p[i].turnaround, p[i].waiting);
    }

    printf("\nAverage Waiting Time: %.2f", avg_wait / n);
    printf("\nAverage Turnaround Time: %.2f\n", avg_turn / n);
    return 0;
}
```

Output:

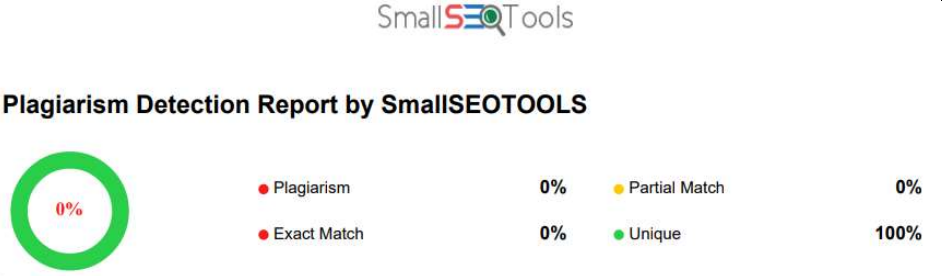
```
input
Enter number of processes: 4
Enter burst time for each process:
P1: 5
P2: 4
P3: 7
P4: 2

Process Burst Time      Waiting Time    Turnaround Time
P4              2               0               2
P2              4               2               6
P1              5               6               11
P3              7               11              18

Average Waiting Time: 4.75
Average Turnaround Time: 9.25


...Program finished with exit code 0
Press ENTER to exit console.
```

Output Analysis	The program successfully simulates both Non-Preemptive and Preemptive CPU Scheduling Algorithms. For Non-Preemptive scheduling (e.g., FCFS or SJF), each process executes completely before the next one begins. For Preemptive scheduling (e.g., SRTF), processes are dynamically interrupted when a shorter burst process arrives. The program accurately computes and displays Waiting Time, Turnaround

	Time, and Completion Time for each process, along with their averages. Results verify that Preemptive algorithms generally reduce waiting and turnaround time for shorter processes compared to Non-Preemptive ones.
Link of student Github profile where lab assignment has been uploaded	"https://github.com/Bhushan-Tayade/YCCN-23071391.git"
Conclusion	The simulation of CPU Scheduling Algorithms was successfully implemented using C. Both Non-Preemptive and Preemptive scheduling methods were analyzed through process execution order and timing metrics. The experiment demonstrated the efficiency and trade-offs between different scheduling techniques — Non-Preemptive methods are simpler but less flexible, while Preemptive methods offer better response and fairness. This practical reinforced understanding of how operating systems manage CPU resources to improve overall system performance and process handling efficiency.
Plag Report (Similarity index < 12%)	
Date	29-10-2025