Nagar Yuwak Shikshan Sanstha's
Yeshwantrao Chavan College of Engineering
(An Autonomous Institution affiliated to Rashtrasant Tukadoji Maharaj Nagpur University)
HingnaRoad,Wanadongri, Nagpur - 441110
NAAC A++
Ph.: 07104-237919,234623,329249,329250Fax:07104-232376,Website: www.ycce.edu

Visionofthe 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: To enhance understanding of operating system concepts by simulating CPU scheduling algorithms and improving analytical skills in process management.	• **Mission:** To provide hands-on experience in implementing non-preemptive and preemptive CPU scheduling techniques, analyze their performance, and develop problem-solving abilities for efficient resource utilization in real-world systems.

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 PEO3	**Core Competence Breadth**	**E: Environment (Learning Environment)**	
PEO4	**Professionalism**	**P: Professionalism**	
PEO5	**Learning**	**C: Core Competence**	
	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
(Signature and Date in Handwritten)

Nagar Yuwak Shikshan Sanstha's
Yeshwantrao Chavan College of Engineering
(An Autonomous Institution affiliated to Rashtrasant Tukadoji Maharaj Nagpur University)
HingnaRoad,Wanadongri, Nagpur - 441110
NAAC A++
Ph.: 07104-237919,234623,329249,329250Fax:07104-232376,Website: www.ycce.edu

DepartmentofComputerTechnologyB.TechinComputerScienceandEngineering(IOT)

Visionofthe 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)**	**CourseName**	**Operating System**
Semester	**5**	**CourseCode**	**23IOT1504**
RollNo	**52**	**Name ofStudent**	**Parth Bhedurkar**

Practical Number	**4**
Course Outcome	**CO2:** Analyse operating system functionalities utilizing system calls, thread programming, and **process scheduling algorithms**. **CO3:** Apply synchronization primitives to implement efficient scheduling and avoid performance bottlenecks. **CO4:** Simulate and evaluate CPU scheduling strategies along with other algorithms such as disk scheduling, memory allocation, and page replacement.
Aim	Write a simulator program CPU scheduling algorithms.
Problem Definition	A. Non pre-emptive CPU scheduling algorithm(Any One). B. Pre-emptive CPU scheduling algorithm.
Theory (100 words)	This practical focuses on the study and implementation of CPU scheduling algorithms, which are an integral part of operating system design. CPU scheduling is the method by which the operating system decides the sequence in which processes are allocated CPU time. When multiple processes are waiting to be executed, an efficient scheduling policy is essential to improve system performance and resource utilization. In this experiment, we implement and analyze two types of scheduling algorithms: Non-preemptive scheduling and Preemptive scheduling. The First Come First Serve (FCFS) algorithm is used as a non-preemptive technique, where processes are executed in the order of their arrival. Once a process begins execution, it runs until completion without interruption. This approach is simple but may lead to longer waiting times for some processes. On the other hand, Round Robin (RR) is implemented as a preemptive algorithm, where each process is assigned a fixed time quantum. The CPU cycles through the

	processes, ensuring that no single process monopolizes the CPU, thus improving fairness and responsiveness. Through this practical, we learn how to calculate important performance metrics such as waiting time, turnaround time, and average response time. Understanding these algorithms provides deeper insights into how operating systems manage process execution to achieve efficiency, fairness, and balanced resource distribution.
Procedure and Execution (100 Words)	**FCFS (Non-preemptive) — Stepwise Procedure :-** 1. Prepare input: list processes as (PID, ArrivalTime, BurstTime). 2. Sort processes by ArrivalTime (tie → by PID). 3. Set current_time = 0. Create arrays to store CT, TAT, WT, and Gantt segments. 4. For each process in sorted order: a. If current_time < ArrivalTime, set current_time = ArrivalTime (CPU idle). b. start_time = current_time c. completion_time = start_time + BurstTime → store CT. d. turnaround = completion_time - ArrivalTime → store TAT. e. waiting = turnaround - BurstTime → store WT. f. Append Gantt segment (PID, start_time, completion_time). g. Update current_time = completion_time. 5. After loop compute averages: avg_TAT = sum(TAT)/n, avg_WT = sum(WT)/n. 6. Print table (PID, AT, BT, CT, TAT, WT), averages, and optional Gantt chart. **Round-Robin (Preemptive) — Stepwise Procedure :-** 1. Prepare input: list processes (PID, ArrivalTime, BurstTime) and time quantum q. 2. Sort by ArrivalTime. Initialize current_time = 0, rem_bt = BurstTime copy, ready_queue = [], arrays for CT/TAT/WT, and Gantt slices list. Set an index i = 0 for arrivals. 3. While there are unfinished processes: a. Add all processes with ArrivalTime <= current_time (from sorted list) to ready_queue and increment i. b. If ready_queue is empty: advance current_time = ArrivalTime of next process and continue. c. Pop idx from front of ready_queue. d. exec = min(q, rem_bt[idx]); record start = current_time; current_time += exec; rem_bt[idx] -= exec. Append Gantt slice (PID, start, current_time).

e. While i has new arrivals with ArrivalTime <= current_time, enqueue them.
f. If rem_bt[idx] > 0 → append idx back to ready_queue; else set CT[idx] = current_time, compute TAT = CT - AT, WT = TAT - BT.

4. Repeat until all rem_bt == 0.
5. Compute averages avg_TAT, avg_WT.
6. Print table and the detailed Gantt timeline showing each time-slice.

Code:

Part– A(FCFS):-

```
import java.util.*;
class Process {
int pid;
float arrivalTime, burstTime, waitingTime, turnaroundTime, completionTime;
Process(int pid, float arrivalTime, float burstTime) {
this.pid = pid;
this.arrivalTime = arrivalTime;
this.burstTime = burstTime;
}
}
public class FCFS_Scheduler {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
System.out.print("Enter number of processes: ");
int n = sc.nextInt();
List<Process> processes = new ArrayList<>();
for (int i = 0; i < n; i++) {
System.out.println("Enter arrival time and burst time for process " + (i + 1) +
":");
float arrival = sc.nextFloat();
float burst = sc.nextFloat();
processes.add(new Process(i + 1, arrival, burst));
}
// Sort by arrival time
processes.sort(Comparator.comparingDouble((p) -> p.arrivalTime));
float currentTime = 0,t_w=0,t_t=0;
for (Process p : processes) {
```

Visionofthe 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.

currentTime = Math.max(currentTime, p.arrivalTime); p.completionTime = currentTime + p.burstTime; p.turnaroundTime = p.completionTime - p.arrivalTime; t_t+=p.turnaroundTime; p.waitingTime = p.turnaroundTime - p.burstTime; t_w+=p.waitingTime; currentTime = p.completionTime; } System.out.println("\nPID\tArrival\tBurst\tWaiting\tTurnaround"); for (Process p : processes) { System.out.printf("%d\t\t%f\t\t%f\t\t%f\t\t%f\n", p.pid, p.arrivalTime, p.burstTime, p.waitingTime, p.turnaroundTime); } float avg_t= (t_t/n); float avg_w=(t_w/n); System.out.println("Avg. turnaround time = "+avg_t+" and avg. waiting time = "+avg_w+"."); } } **Part – B (Round Robin):** importjava.util.Arrays; publicclass RoundRobin {

```
staticvoid roundRobin(int[] burst, int quantum) {
    intn= burst.length;


    int[]remaining = Arrays.copyOf(burst, n);
    int[]wait = new int[n];      // waiting time
    int[]tat = new int[n];       // turnaround time
    int[]finish = new int[n]; // completion time (optional, but useful)

    inttime = 0;                 // current time
    boolean allDone;
    System.out.println("Execution (Gantt-like):");
```

Visionofthe 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.

```
        do{
           allDone = true;
           for (int i = 0; i < n; i++) {

              if (remaining[i] > 0) {
                 allDone=false;//atleastoneprocessstill needs CPU


                 intslice=Math.min(remaining[i],quantum);


                 System.out.print("P"+(i+1)+"("+time + "->" + (time + slice) +
") ");

                 time += slice;
                 remaining[i] -= slice;


                 if (remaining[i]==0) {
                    finish[i] = time;
                 }
              }
           }
        } while (!allDone);
        System.out.println();//newlineaftertheGantt-like log


        int totalWT = 0, totalTAT = 0;
        for (int i = 0; i < n; i++) {
           tat[i]=finish[i];          // sincearrival=0
           wait[i]=tat[i] -burst[i];//TAT -BT
           totalWT+=wait[i];
           totalTAT +=tat[i];
        }
        // Printtable
        System.out.println("\nPN BT WT TAT");
        for (int i = 0; i < n; i++) {

           System.out.printf("P%-3d%-4d%-4d%-4d%n", (i + 1), burst[i],
wait[i], tat[i]);
        }
        System.out.printf("%nAveragewaitingtime=%.2f%n", (totalWT * 1.0) /
```

Visionofthe Department

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

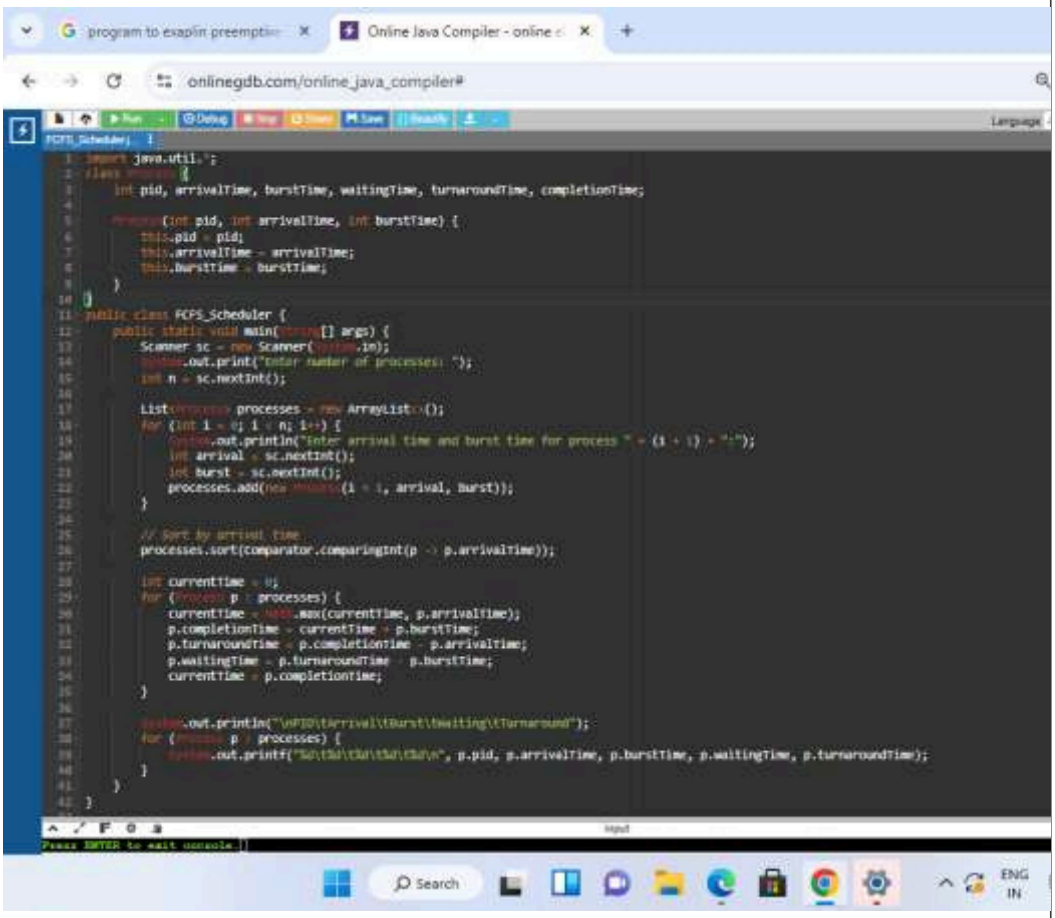
Mission of the Department

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

```
n);
/ n); System.out.printf("Average turn around time = %.2f%n", (totalTAT * 1.0)

    }
    public static void main(String[] args) {

        // Example: 3 processes, only burst times known (all arrive at time 0)
        int[] burst = {10, 5, 8};
        int quantum = 2;
        roundRobin(burst, quantum);

    }
}
```

Output:

Part A:-

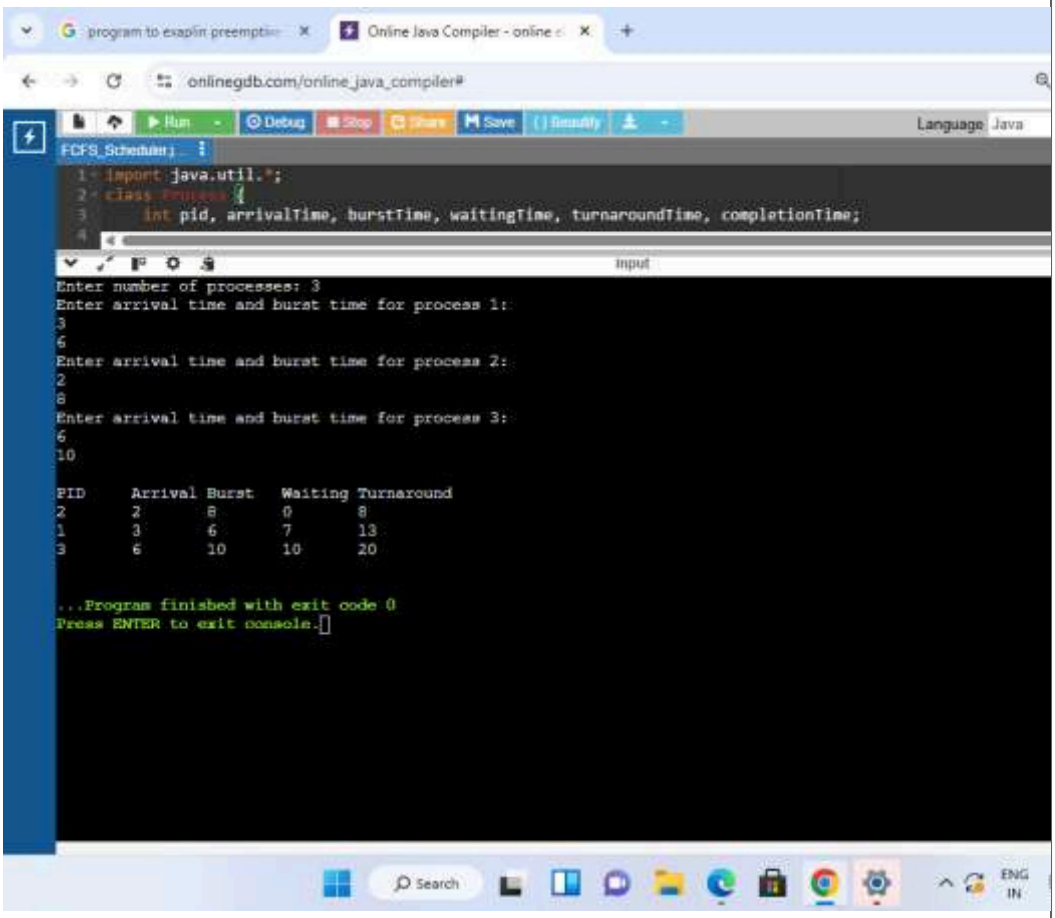


Visionofthe Department

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.

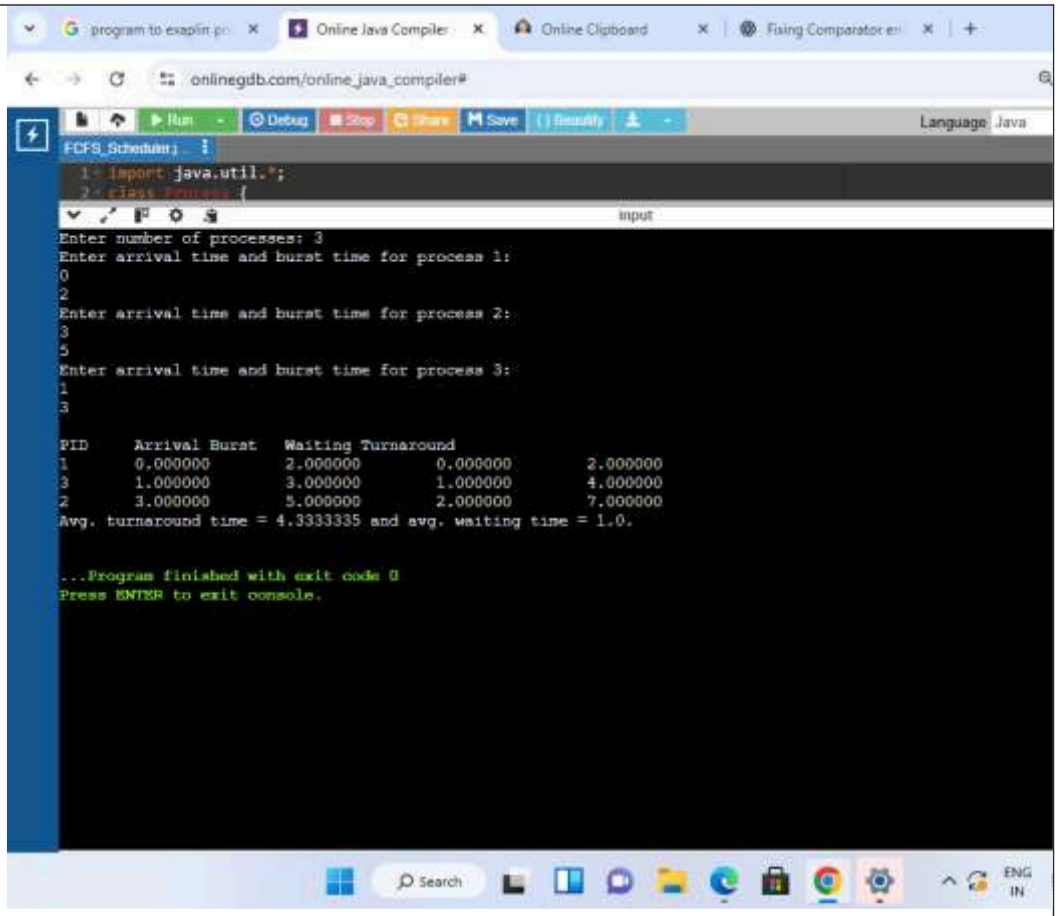


Visionofthe Department

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.



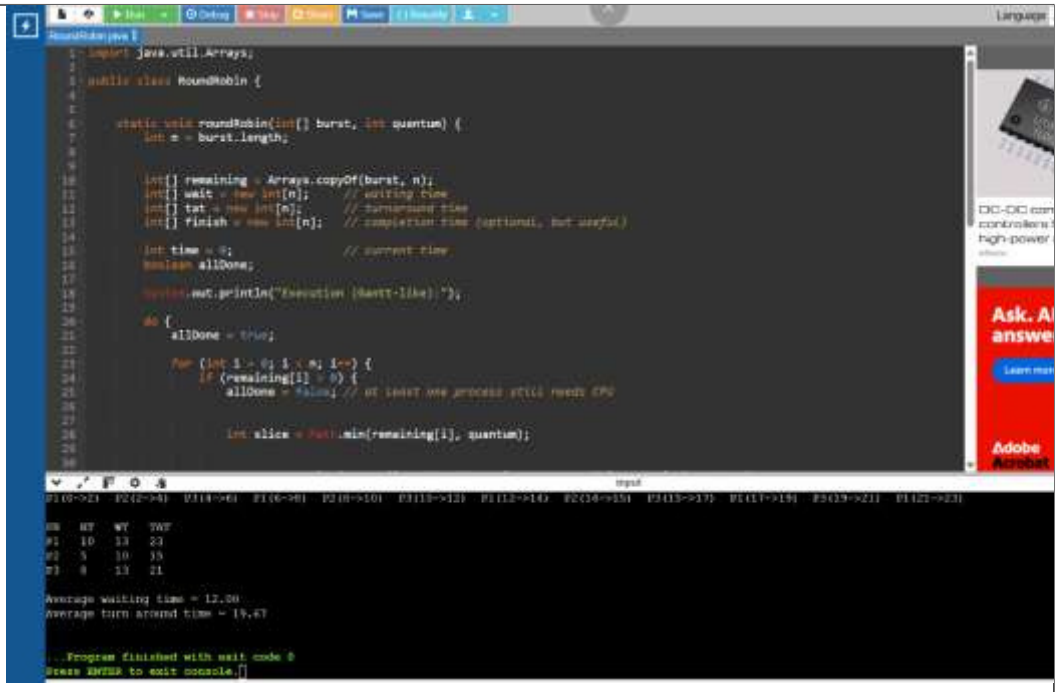
Part B:-

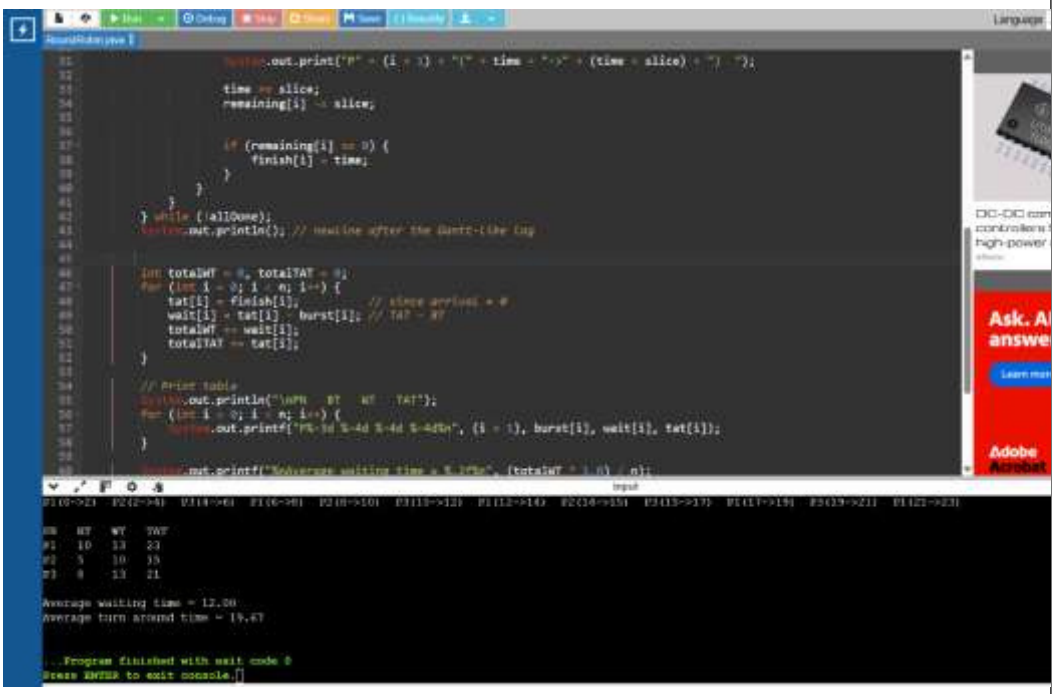
Visionofthe Department

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.



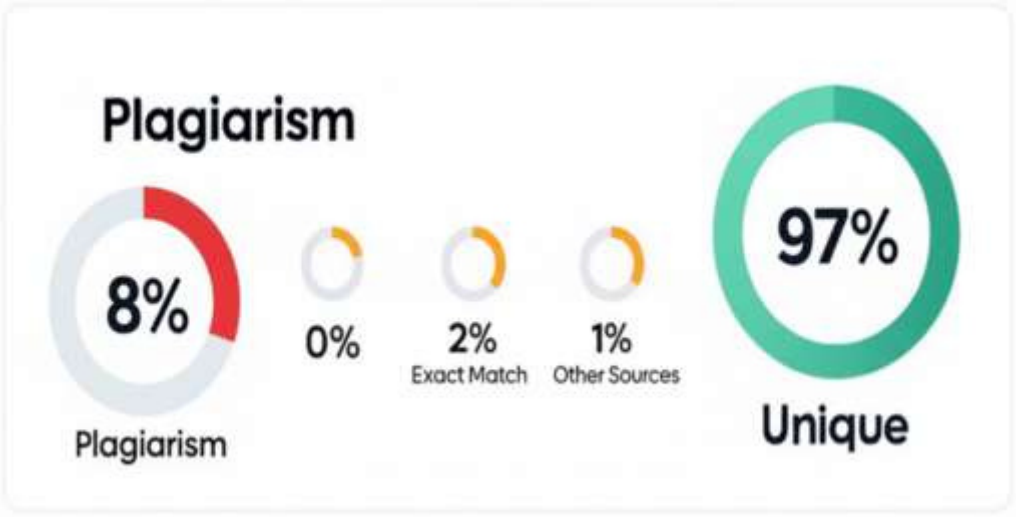


Visionofthe Department

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.

Output Analysis	From the simulation results, we observe that in **FCFS scheduling**, processes are executed in the order of arrival, which leads to simple execution but longer waiting times for processes arriving later (convoy effect). On the other hand, **Round Robin scheduling** distributes CPU time fairly among all processes using a fixed quantum, improving responsiveness and reducing starvation. However, context switching increases and average turnaround time may be higher compared to FCFS. The practical demonstrates how scheduling policies directly affect **waiting time, turnaround time, and fairness** in process execution.
Link of student Github profile where lab assignment has been uploaded	https://github.com/parthbhedurkar
Conclusion	The practical successfully demonstrates the working of **CPU scheduling algorithms**. FCFS is simple but may cause longer waiting times for later processes, while Round Robin improves fairness by sharing CPU time equally among processes. Through this experiment, we learned how scheduling policies affect **waiting time, turnaround time, and overall system performance**.
Plag Report (Similarity index < 12%)	
Date	06/09/2025