THINGS TO KNOW:

1. **Lab report must contain following sections: (order must be maintained)**
   a) **Title /Question**
   b) **Theory: The brief overview of the concept /techniques/syntax/technology used in the program**
   c) **Code: The complete code**
   d) **Output: Screenshot of the output**
2. **Output screen should be captured (use snipping tool), printed and attached in the report.**
3. **Every Source code must include the printing statements to print following information after your main output:**

   Lab No.:

   Name:

   Roll No./Section :

4. **Contents should be written/printed on single side of A4 sized paper.**
5. **Cover page and contents page should be attached in the report appropriately**.
6. **The works must be submitted within specified deadline.**

Contents Page Format

Contents

Lab No.	**Title /Question**	**Submission Date**	**Signature**	**Remarks**
1(a)	This is sample title	2079/03/15		
1(b)	This is another title	2079/03/17		

Remaining Lab Works (OS)

2. **Process /Thread Creation and Termination**

   (Write C programs and run in Linux using gcc)

   a) WAP in C to demonstrate the process creation and termination in Linux.

   b) WAP in C to demonstrate the thread creation and termination in Linux.

3. **Simulation of IPC Techniques**

   a. WAP in C to simulate shared memory concept for IPC.

   b. WAP in C to simulate message passing concept for IPC.

4. **Simulation of Process Scheduling Algorithms**

(Take the inputs like number of processes, arrival time, burst time and priority values from users, show the values in tabular form and display average turnaround time, average waiting time and average response time.)

   a. WAP in C to simulate FCFS CPU Scheduling Algorithm

   b. WAP in C to simulate SJF CPU Scheduling Algorithm

   c. WAP in C to simulate SRTF CPU Scheduling Algorithm

   d. WAP in C to simulate Round Robin CPU Scheduling Algorithm

   e. WAP in C to simulate Non-Preemptive Priority Scheduling Algorithm

   f. WAP in C to simulate Preemptive Priority Scheduling Algorithm

5. **Simulation of Deadlock Avoidance and Deadlock Detection Algorithms**

   (Output screen should contain both positive and negative instances i.e. safe and unsafe both)

   a. WAP to implement Bankers Algorithm for multiple type of resources to decide safe/unsafe state.

   b. WAP for deadlock detection in the system having multiple type of resources. The program should list the deadlocked process in case of deadlock detection results true)

6. **Simulation of Page Replacement Algorithms**

   (Take the reference string from user)

   a. WAP in C to simulate FIFO Page Replacement Algorithm

   b. WAP in C to simulate Optimal Page Replacement Algorithm

c. WAP in C to simulate LRU Page Replacement Algorithm
d. WAP in C to simulate Second Chance Page Replacement Algorithm
e. WAP in C to simulate LFU Page Replacement Algorithm

7. Simulation of disk scheduling algorithms

(Take the access requests from user)

a. WAP to simulate FCFS Disk Scheduling Algorithm
b. WAP to simulate SSTF Disk Scheduling Algorithm
c. WAP to simulate SCAN Disk Scheduling Algorithm
d. WAP to simulate C-SCAN Disk Scheduling Algorithm
e. WAP to simulate LOOK Disk Scheduling Algorithm
f. WAP to simulate C-LOOK Disk Scheduling Algorithm