1. **Problem Statement 1: Two-Pass Assembler**

Design suitable Data structures and implement Pass-I and Pass-II of a two-pass assembler for pseudo-machine. Implementation should consist of a few instructions from each category and few assembler directives. The output of Pass-I (intermediate code file and symbol table) should be input for Pass-II.

2. **Problem Statement 2: Two-Pass Macro Processor**

Design suitable data structures and implement Pass-I and Pass-II of a two-pass macroprocessor. The output of Pass-I (MNT, MDT and intermediate code file without any macro definitions) should be input for Pass-II.

3. **Problem Statement 3: First Come First Serve (FCFS) CPU Scheduling**

Write a program to simulate the **First Come First Serve (FCFS)** CPU scheduling algorithm. The program should accept process details such as **Process ID, Arrival Time, and Burst Time** and compute the **Waiting Time** and **Turnaround Time** for each process. Display the **Gantt chart**, **average waiting time**, and **average turnaround time**.

4. **Problem Statement 4: Shortest Job First (SJF – Preemptive) Scheduling**

Write a program to simulate the **Shortest Job First (SJF – Preemptive)** CPU scheduling algorithm. The program should calculate and display the **order of execution**, **waiting time**, **turnaround time**, and their averages for all processes.

5. **Problem Statement 5: Priority Scheduling (Non-Preemptive)**

Write a program to simulate the **Priority Scheduling (Non-Preemptive)** algorithm. Each process should have an associated **priority value**, and the scheduler should select the process with the **highest priority** for execution next. Compute and display the **waiting time**, **turnaround time**, and **average times** for all processes.

6. **Problem Statement 6: Round Robin (RR) Scheduling**

Write a program to simulate the **Round Robin (Preemptive)** CPU scheduling algorithm. The program should take **time quantum** as input and schedule processes in a cyclic order. Display the **Gantt chart**, **waiting time**, **turnaround time**, and **average values** for all processes.

7. **Problem Statement 7: Memory Allocation – First Fit**

Write a program to simulate **First Fit** memory allocation strategy. The program should allocate each process to the first available memory block that is large enough to accommodate it. Display the **memory allocation table** and identify any **unused or fragmented memory**.

8. **Problem Statement 8: Memory Allocation – Best Fit**

Write a program to simulate **Best Fit** memory allocation strategy. The program should allocate each process to the smallest available block that can hold it. Display the **final allocation** and show **internal fragmentation** if any.

9. **Problem Statement 9: Memory Allocation – Next Fit**

Write a program to simulate **Next Fit** memory allocation strategy. The program should continue searching for the next suitable memory block from the last allocated position instead of starting from the beginning. Display the **memory allocation table** and **fragmentation details**.

10. **Problem Statement 10: Memory Allocation – Worst Fit**

Write a program to simulate **Worst Fit** memory allocation strategy. The program should allocate each process to the **largest available memory block**. Display the **memory allocation results** and any **unused space**.

11. **Problem Statement 11: Page Replacement – FIFO**

Write a program to simulate the **First In First Out (FIFO)** page replacement algorithm. The program should accept a **page reference string** and **number of frames** as input, simulate the process of page replacement, and display the **total number of page faults**.

12. **Problem Statement 12: Page Replacement – LRU**

Write a program to simulate the **Least Recently Used (LRU)** page replacement algorithm. The program should display the **frame contents after each page reference** and the **total number of page faults**.

13. **Problem Statement 13: Page Replacement – Optimal**

Write a program to simulate the **Optimal Page Replacement** algorithm. The program should replace the page that **will not be used for the longest period of time** in the future and display the **page replacement steps** and **page fault count**.