Problem 1. (30 Points) Demand page with pre-fetch problem. Assume that \{PA, PB\} are two programs. Assume that \{P1, P2, P3\} are in the RQ before t=0 and \{P1, P2\} are executing PA while P3 is executing PB. Assume that when PA is executed, different page reference strings will be recorded as the input data changed. Say, the page reference string P1 recorded as \{4, 5, 4, 5, 6\} and P2 as \{4, 4, 5, 5, 6\}. and where the underlined 5 means the first instruction of page #5 will forks a new process which is executing PB and we assume that page #7 is not existed. When PB is executed, a reference string will be recorded as: \{4, 4, 4\} and we assume that page #5 is not existed. Assume that it takes 3 ticks to handle each page fault and 2 ticks to handle additional fetched page, and I/O device uses a FCFS strategy. Draw a Gantt chart illustrating the scheduling if a round-robin scheduling with a time quantum of 3 ticks is used. Assume that we can ignore the page replacement issue. Show your RQ, I/O Q, BQ, page tables and working sets as references. Mark on the chart when a new process is arrived or finished.
Problem 2. (10 Points) It is said that a dead lock is the result of a bad timing. Use an example to show the reason why you are agree or disagree with this declaration. What’s the simplest method to prevent dead lock? Why it is working and what is the major concern problem with this solution? Use a RAG to justify your answer.

![Diagram](image)

Problem 3. (20 Points) Demand page problem. Assume that PA is a program. Assume that when this program is executed, the following page reference string will be recorded: \{3, 3, 4, 4, 5\}. Assume that the last instruction of page #3 will requests an I/O. Assume that the execution of each page takes 1 tick, and each I/O needs 3 ticks to be handled with a FCFS strategy. Assume that there are three processes \{P1, P2, P3\} in the RQ before t=0, and all of them are executing PA. Draw a Gantt chart if a RR scheduling with a time quantum of 2 ticks is used and assume that each process has a large enough working set.

![Diagram](image)