Please note that for all tasks mentioned below, your solution will be considered acceptable if it employs CPU parallelization and synchronization techniques using the standard C++ library. Your application must also fulfil all the requirements outlined in the Assessment Brief Document on MLS. Additionally, it must meet three specific criteria for each idea as described below.

Idea 1: Upgrading the Elevator using CVs.

Task 1: Upgrade the elevator simulation code from Lab3 using condition variables (or other synchronisation approaches) to better model passenger behaviour in two scenarios: waiting for the elevator and riding inside it. Currently, the simulation uses while loops to simulate passengers pressing the call button every 100 ms because they're unaware of the elevator's position relative to their floor. This approach inaccurately represents how passengers interact with elevators, as they typically press the call button once and wait. Similarly, while inside the elevator, passengers are simulated as continuously pressing a button to indicate their destination floor, unaware of the elevator passing their desired floor. This behaviour differs from real-life scenarios where passengers select their floor once. This is not like a real-life scenario. - Done

Task 2: In addition to the deadlock problem, you addressed in the practical, the original elevator code intentionally contains another logical bug that allows passengers to board and be delivered to floors other than the current one. Identify the source of this bug and correct it.

Hint 1: In this simulation, a certain set of activities requires atomicity.

Hint 2: In certain cases, some synchronization techniques may perform better than others.

Task 3: Allow passengers to make requests dynamically at runtime instead of predefining a fixed number of passengers with predetermined start and destination floors. This could involve creating a user interface or a command-line input method where new passenger requests can be added while the simulation is running. Alternatively, you can create another thread to act as a passenger generator, which continuously creates a stream of passengers. This simulates a more realistic scenario where people arrive and call the elevator at different times/floors.