- Since the plane matrix, the number of occupied seats (or remaining seats), and the turn variable will be shared by child threads and the main thread, I declared them as global variables.

- The job of the child threads is the same, therefore I wrote a single function for the reservation purposes and send the id of the thread as a parameter to the function that would be executed by both the threads concurrently.

- The child threads for the agencies would run while there are still seats that are not reserved by any of the agencies. The outer while loop in the agency_function works for that purpose. When the while loop terminates, both of the threads would join the main thread. Inside the while loop, there is busy waiting implemented with a while loop again.
    - If the current turn is not equal to the id of the thread which is scheduled at a particular time, then it busy waits.
    - If the current turn is equal to the id of the executing thread, then:
        - It prints a message that now it is in the critical region
        - Generates a random number, convert that random number into a row and column values of the plane matrix
        - Checks if the matrix element corresponding to the random number is reserved or not
            - If reserved, then there is no overbooking
                - Prints the message of leaving the critical region
                - Switching the turn
            - If the corresponding seat is not reserved then it reserves
                - Increment the occupiedSeats variable which is the loop variable
                - Prints a message about the reservation information
                - Prints the message of leaving the critical region

- For all 100 seats, this process is repeated and threads execute until all the seats are reserved. When the occupied seats become 100, both of the threads terminate the loop and join the main thread. Then what we do is simply printing the plane and terminating the main thread at the end.