Name: Mubasshir Al Shahriar (Nihal)

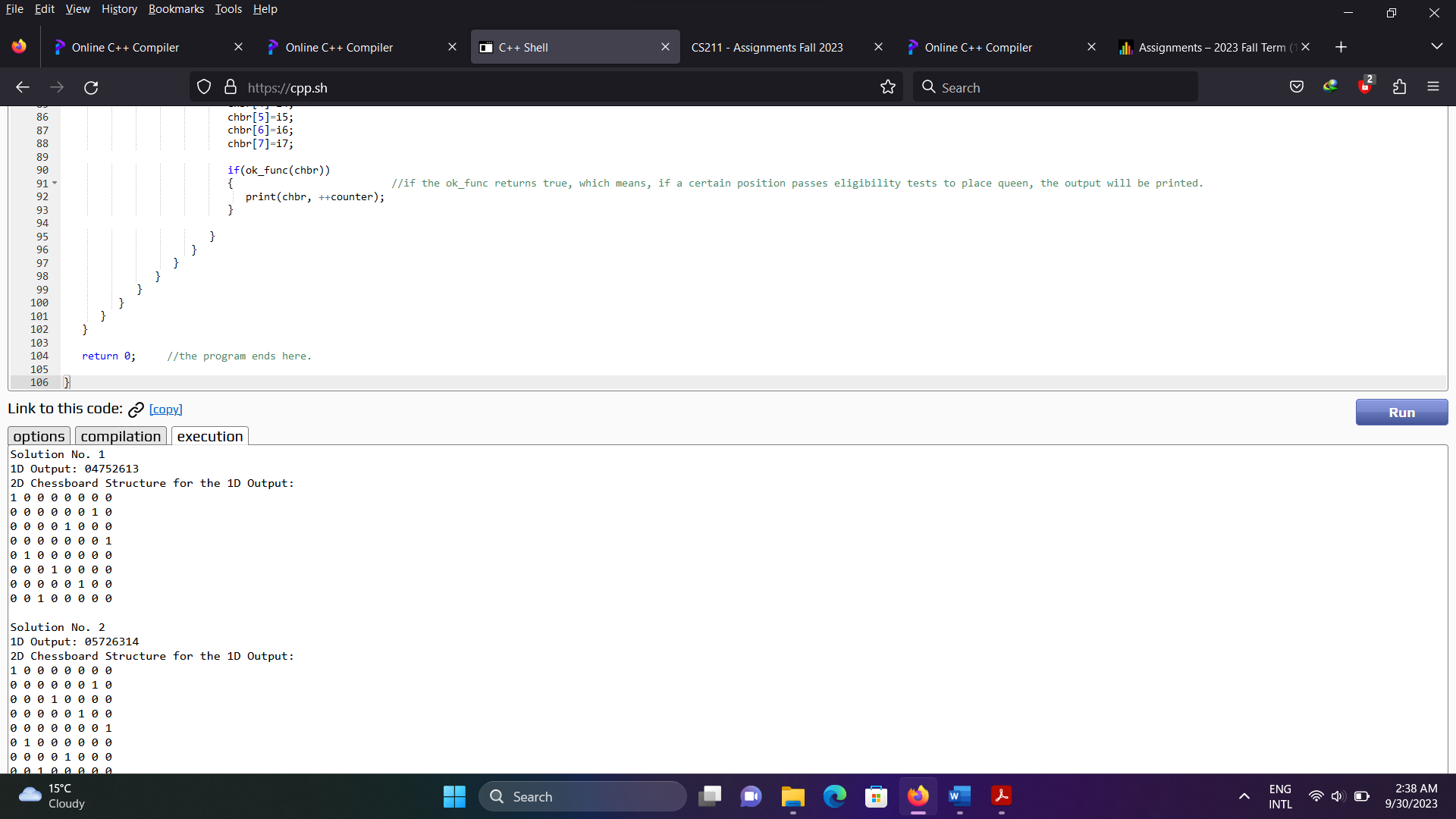
Assignment: Complete the "dumb" 8 queens program that we looked at in lecture. Use the 1 dimensional array

Short Report/Summary:

* Created a function named “ok\_func” that will run the eligibility test which will detect if the position is suitable or not to put a queen by checking parallel rows, up diagonal and down diagonal.
* If the position is suitable, that the function will return a true value which will let the program in main function to print that value as a chessboard structure. If it fails any of the 3 conditions in the if statement, it will return false and if it returns false, it won’t print that result**. In simple words, if the position is fine to put the queen, it will be printed.**
* Then presented a void function named “print” to print the values as a 2D structure of chessboard. **In simple words, this function will print the desired chessboard with 8 queens placed in correct places.**
* Inside main function, using 8 nested for loops, it assigns all the possible configurations (8^8 configurations) of the chessboard with 8 queens. Then it calls the ok\_func function to check which configurations among them are good to print, and the configurations which result in returning true in ok\_func function gets printed**. In a simple word, among all the possible configurations of the chessboard, only the configurations with 8 queens placed in valid positions are printed. Thus, we can find our desired solutions of 8 queens in the chessboard.**

Comment: This is my algorithm to place 8 queens in a chessboard so none of them threat each other using 1D array and not using any goto statement. This is much different from the one where we used goto statement. Because, at that previous version, the program checked specific positions, for instance, whenever it found a queen in the same row of a column, it didn’t checked the rest of the row, thus that became more efficient. But as this one is the dumb version, here we assigned all the possible configurations (8^8 configurations) and checked all of those possible configurations with ok function and finally printed only the valid configurations/solutions. So, this version must be lot more inefficient than the previous one. I compiled and ran the program, it ran correctly.

Screenshots of Output:



**A computer screen with a white screen

Description automatically generated**