Explanation for N-Queens Problem

## ****1. Introduction to the N-Queens Problem****

The N-Queens problem is a classic combinatorial problem where the objective is to place **N queens** on an **N×N chessboard** such that no two queens attack each other. This means:

* No two queens can be in the same row.
* No two queens can be in the same column.
* No two queens can be on the same diagonal.

## ****2. Code Explanation****

The provided Python code allows users to **interactively place queens** on an N×N chessboard while checking the validity of each placement.

### ****is\_valid(board, row, col, n)****

This function checks if placing a queen at position (row, col) is valid:

* It verifies that there is no queen in the same **column**.
* It checks the **upper left diagonal**.
* It checks the **upper right diagonal**.
* Returns True if the position is safe, otherwise False.

### ****print\_board(board)****

This function prints the chessboard in a **readable format**.

### ****is\_solution\_correct(board, n)****

This function verifies if the final board is a valid solution:

* Extracts positions of all queens.
* Checks if any two queens are in the same diagonal.
* Returns True if the placement is valid, otherwise False.

### ****n\_queens() (Main function)****

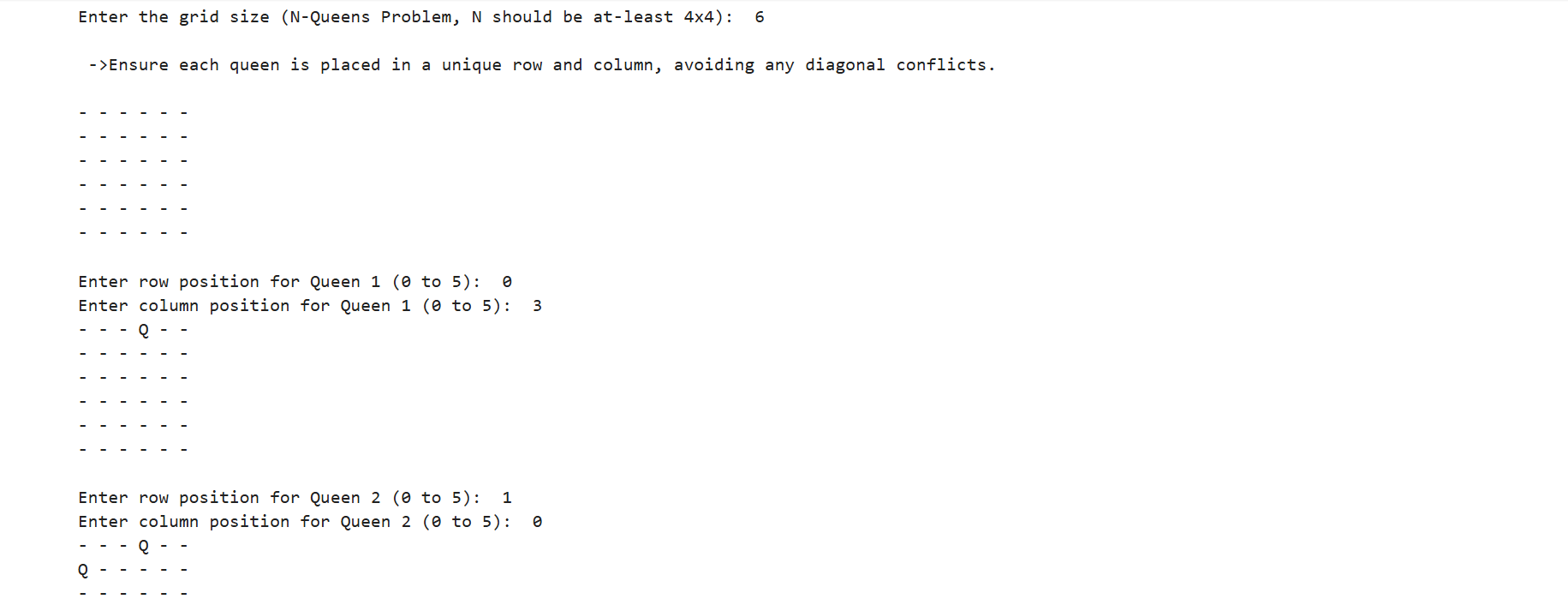
This function manages the interactive placement of queens:

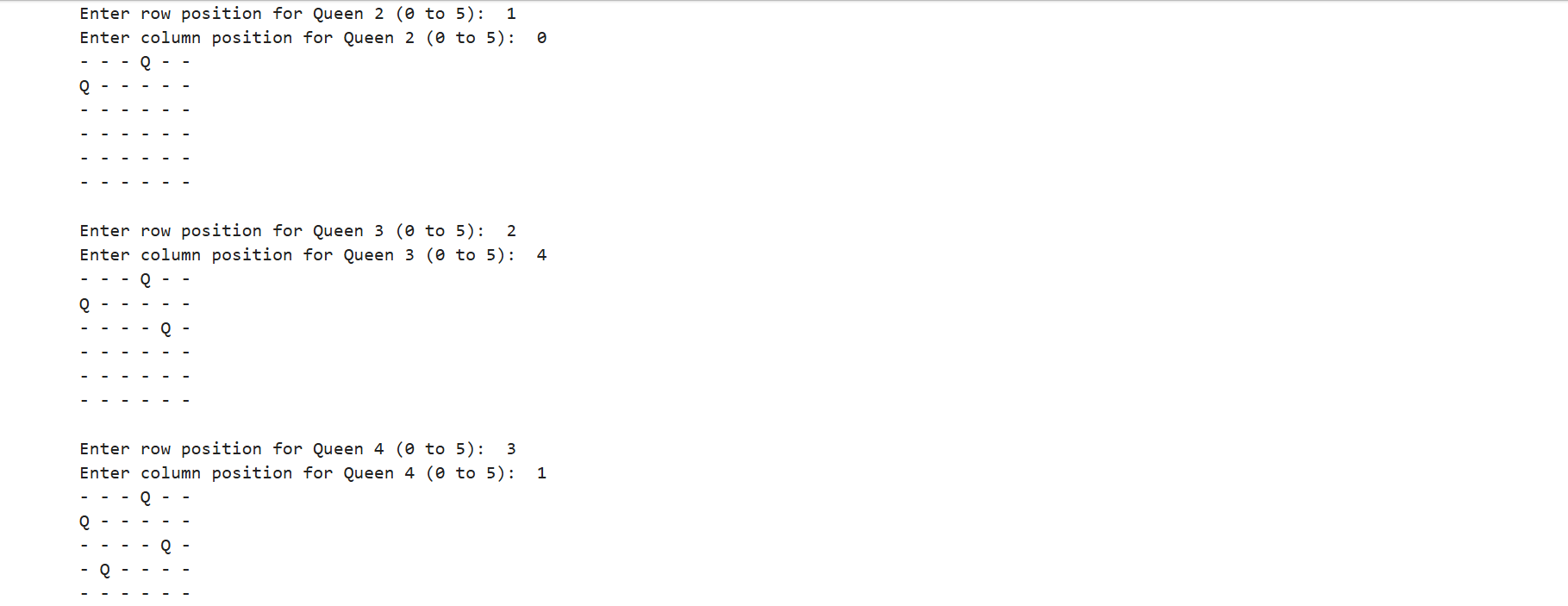
* Asks the user for the grid size n (must be at least 4x4).
* Creates an empty board.
* Guides the user to place N queens while checking validity.
* After placement, verifies if the solution is correct.
* If incorrect, allows retrying or exiting.

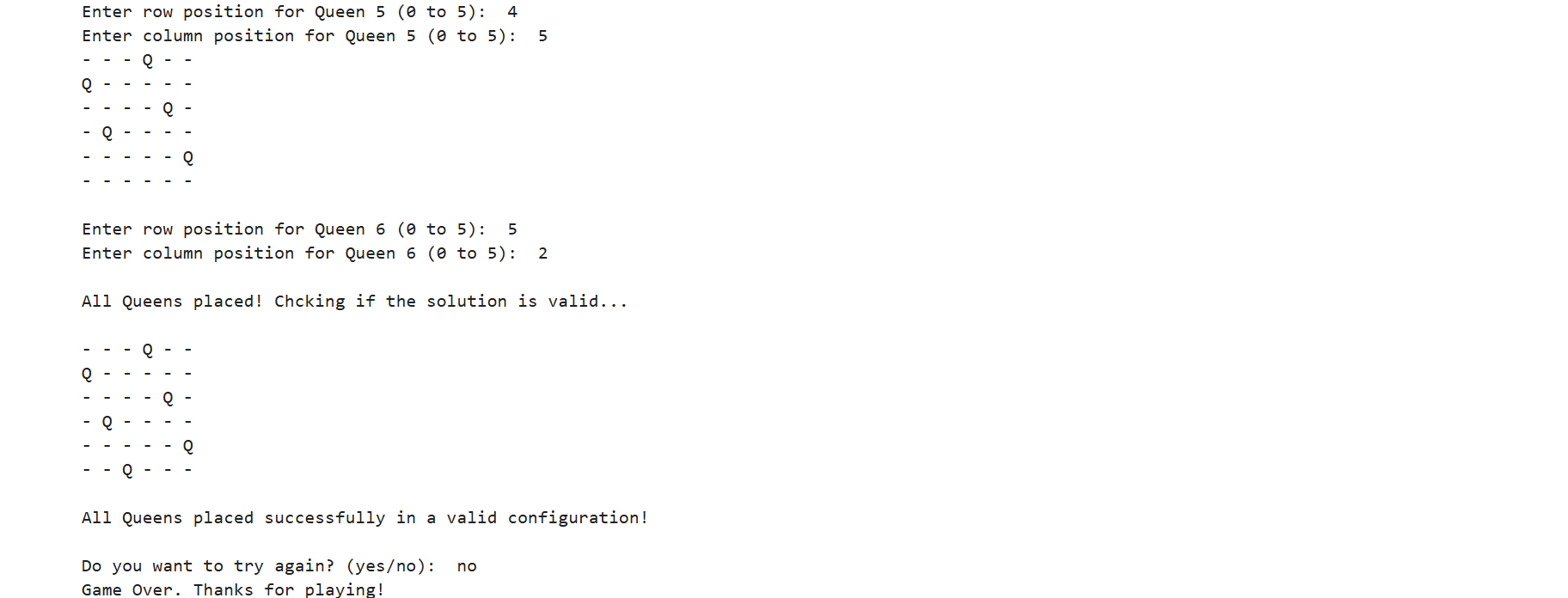
## ****3. How the Code Works?****

1. The user is asked to enter a board size n (at least 4x4).
2. The program initializes an empty N×N board filled with '-'.
3. The user is asked to place N queens one by one.
4. Each move is validated using is\_valid(board, row, col, n).
5. If all N queens are placed successfully, the program checks if the placement is correct.
6. If valid, the solution is confirmed; otherwise, the user is prompted to retry.

## ****4. Screenshot of Output****







## ****5. Accuracy & Validity****

* The program ensures correctness by verifying that no queens attack each other.
* It prevents invalid moves by checking rows, columns, and diagonals.
* The final validation function ensures correctness of the solution before confirming success.

## ****6. Conclusion****

This implementation of the **N-Queens Problem** is interactive, allowing users to understand and solve the problem manually. The combination of **validation functions** ensures that only correct solutions are accepted.