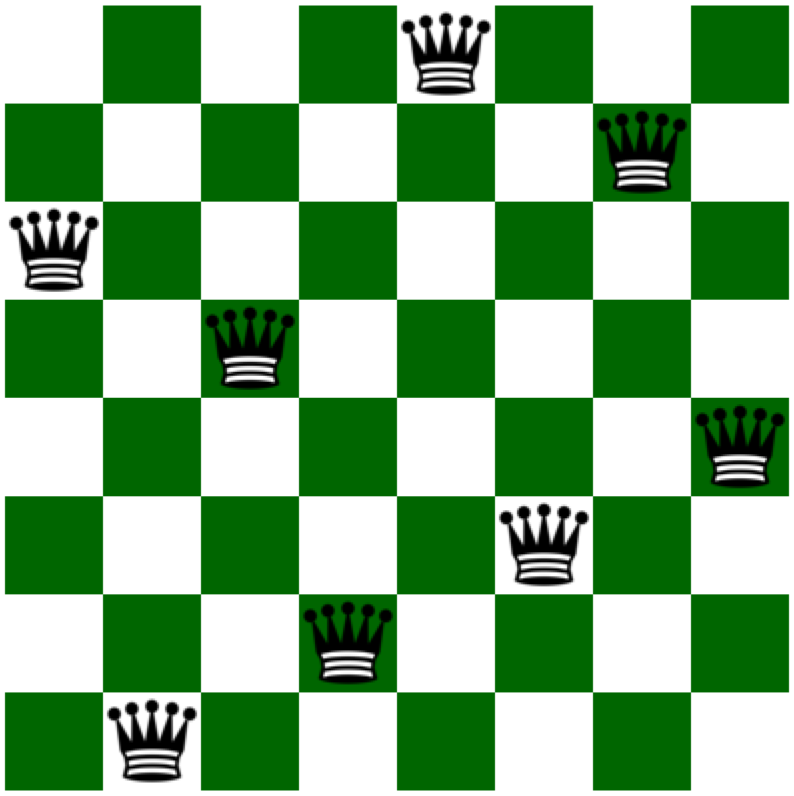
Local Search using Hill climbing.

The **N** Queen is the problem of placing **N** chess queens on an **N×N** chessboard so that no two queens attack each other. For example, the following is a solution for **8** Queen problem.



Input: *N = 4*
Output:

```
0 1 0 0
0 0 0 1
1 0 0 0
0 0 1 0
```

Explanation:
The Position of queens are:
1 – {1, 2}
2 – {2, 4}
3 – {3, 1}
4 – {4, 3}
As we can see that we have placed all 4 queens in a way that no two queens are attacking each other. So, the output is correct

Input: *N = 8*
Output:
```
0 0 0 0 0 0 1 0
0 1 0 0 0 0 0 0
0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0
1 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 1
0 0 0 0 1 0 0 0
```

Approach: The idea is to use Hill Climbing Algorithm. While there are algorithms like Backtracking to solve N Queen problem, let's take an AI approach in solving the problem. It's obvious that AI does not guarantee a globally correct solution all the time, but it has quite a good success rate of about 97% which is not bad. A description of the notions of all terminologies used in the problem will be given and are as follows:

Notion of a State – A state here in this context is any configuration of the N queens on the N X N board. Also, in order to reduce the search space let's add an additional constraint that there can only be a single queen in a particular column. A state in the program is implemented using an array of length N, such that if **state[i]=j** then there is a queen at column index **i** and row index **j**.

Notion of Neighbor s – Neighbor s of a state are other states with board configuration that differ from the current state's board configuration with respect to the position of only a single queen. This queen that differs a state from its neighbor may be displaced anywhere in the same column.

Optimization function or Objective function – We know that local search is an optimization algorithm that searches the local space to optimize a function that takes the state as input and gives some value as an output. The value of the objective function of a state here in this context is the number of pairs of queens attacking each other. Our goal here is to find a state with the minimum objective value. This function has a maximum value of NC2 and a minimum value of 0.

Algorithm:
1. Start with a random state (i.e, a random configuration of the board).
2. Scan through all possible neighbor s of the current state and jump to the neighbor with the highest objective value, if found any. If there does not exist, a neighbor, with objective strictly higher than the current state but there exists one with equal then jump to any random neighbor (escaping shoulder and/or local optimum).

3. Repeat step 2, until a state whose objective is strictly higher than all its neighbor 's objectives, is found and then go to step 4.
4. The state thus found after the local search is either the local optimum or the global optimum. There is no way of escaping local optima but adding a random neighbor or a random restart each time a local optimum is encountered increases the chances of achieving global optimum (the solution to our problem).
5. Output the state and return.

It is easily visible that the global optimum in our case is 0 since it is the minimum number of pairs of queens that can attack each other. Also, the random restart has a higher chance of achieving global optimum, but we still use random neighbor because our problem of N queens does not have a high number of local optima and random neighbor is faster than random restart.

Conclusion:

- **Random Neighbor** escapes shoulders but only has a little chance of escaping local optima.
- **Random Restart** both escapes shoulders and has a high chance of escaping local optima.

Task

Implement the above algorithm to solve the N Queen Problem. A helping code for this is already provided in the Hill Climbing AI.ipynb file. Definitions for all functions are already provided. You Need to write the missing codes at the provided place.

Note: If you want to use another implementation and do the code from scratch, feel free to do it, or modify any function according to your understanding. The objective is to solve the N Queen problem with Hill climbing. Feel free to use any version of implementing/ functions you want.