## Code:

```
#include <bits/stdc++.h>
#define N 4
using namespace std;


void printSolution(int board[N][N])
{
	for (int i = 0; i < N; i++) {
		for (int j = 0; j < N; j++)
		if(board[i][j])
			cout << "Q ";
		else cout<<". ";
		printf("\n");
	}
}

bool isSafe(int board[N][N], int row, int col)
{
	int i, j;

	for (i = 0; i < col; i++)
		if (board[row][i])
			return false;

	for (i = row, j = col; i >= 0 && j >= 0; i--,
j--)
		if (board[i][j])
			return false;

	for (i = row, j = col; j >= 0 && i < N; i++,
j--)
		if (board[i][j])
			return false;

	return true;
}

bool solveNQueen(int board[N][N], int col){
	if (col >= N)
		return true;

	for (int i = 0; i < N; i++) {
		if (isSafe(board, i, col)) {
			board[i][col] = 1;
			if (solveNQueen(board, col
+ 1)){
		cout<<"\nValid configuration found:\
n";
		printSolution(board);
				return true;
		}
			board[i][col] = 0;
		}
	}
   cout << "\nInvalid placement of queen: \n";
   printSolution(board);

	return false;
}

int main(){

   int board[N][N];
   for(int i=0; i<N; i++)
      for(int j=0; j<N; j++)
         board[i][j] = 0;

   solveNQueen(board, 0);

   return 0;
}
```

## Output:

somesh4545 ➜ /workspaces/TE-Labs/DAA (main) $ g++ n-queen.cpp && ./a.out

Invalid placement of queen:
Q . . .
. . . .
. Q . .
. . . .

Invalid placement of queen:
Q . . .
. . Q .
. . . .
. Q . .

Invalid placement of queen:
Q . . .
. . . .
. . . .
. Q . .

Invalid placement of queen:
Q . . .
. . . .
. . . .
. . . .

Valid configuration found:
. . Q .
Q . . .
. . . Q
. Q . .

Valid configuration found:
. . Q .
Q . . .
. . . Q
. Q . .

Valid configuration found:
. . Q .
Q . . .
. . . Q
. Q . .

Valid configuration found:
. . Q .
Q . . .
. . . Q
. Q . .