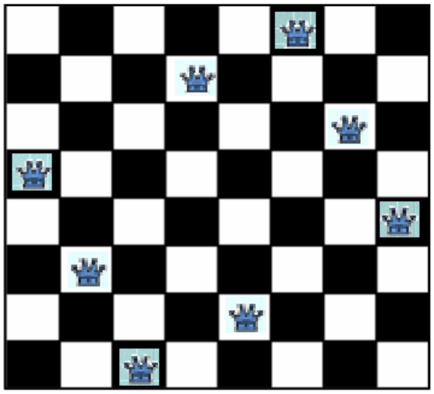
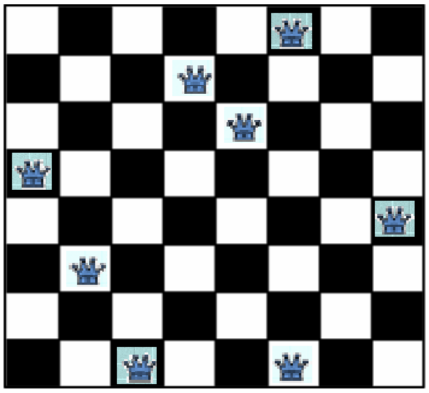
Lab 2: Problem Solving by Searching: N Queens Problem

Problem Statement: The problem is to place 8 queens on a chessboard so that no two queens are in the same row, column or diagonal.

The picture below on the left shows a solution of the 8-queens problem. The picture on the right is not a correct solution, because some of the queens are attacking each other.





Problem Formulation:

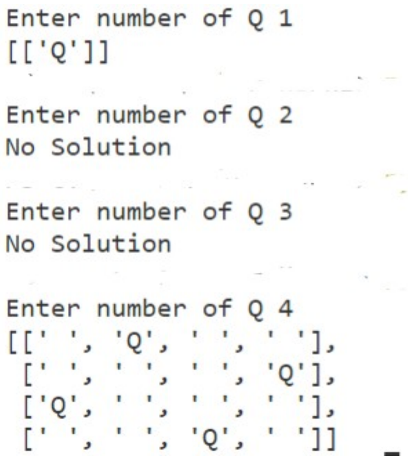
- **State Space:** Any arrangement of **k** queens in the first k rows such that none are attacked
- **Initial state:** 0 queens on the board
- **Successor function:** Add a queen to the (k+1)th row so that none are attacked.
- **Goal test:** 8 queens on the board, none are attacked

Source Code:

```
import pprint

def isSafe(board, x, y, n):
  #Checking whether the column is filled
  for row in range(x):
    if(board[row][y] == 'Q'):
      return False

  #Checking for top left diagonals are filled
  row = x
  col = y
  while(row>=0 and col>=0):
    if(board[row][col] == 'Q'):
      return False
    row -= 1
    col -= 1

  #Checking for top right diagonals are filled
  row = x
  col = y
  while(row>=0 and col<n):
    if(board[row][col] == 'Q'):
      return False
    row -= 1
    col += 1
  #return True if all the aforementioned tests is passed
  return True

def nQueen(board, x, n):
  #if we have successfully placed n queens return True
  if(x>=n):
    return True
  #iterate through columns for each row
  for col in range(n):
    #if the particular position is safe then place that queen
    if(isSafe(board,x,col,n)):
      board[x][col] = 'Q'
      #if the next queen cannot be placed then backtrack
      if(nQueen(board,x+1,n)):
        return True
      board[x][col] = ' '
  return False
```

```
n = int(input("Enter number of Q "))
board = [[' ']*n for i in range(n)]
if(nQueen(board,0,n)):
  pprint.pprint(board)
else:
  print("No Solution")
```

Output:



```
Enter number of Q 1
[['Q']]

Enter number of Q 2
No Solution

Enter number of Q 3
No Solution

Enter number of Q 4
[[' ', 'Q', ' ', ' '],
 [' ', ' ', ' ', 'Q'],
 ['Q', ' ', ' ', ' '],
 [' ', ' ', 'Q', ' ']]
```