Program 1: write a program in prolog to solve N X N queen problem.

Code:
```
queen(N,Queens):-
range(1, N, Rows),
permutation(Rows, Queens),
safe(Queens).
range(Start,End, [Start|Rest]) :-
Start =<End,
NewStart is Start +1,
range(NewStart, End, Rest).
range(End, End, []).
safe([]).
safe([Queen|Queens]):-
safe(Queens, Queen, 1),
safe(Queens).
safe([], _, _).
safe([OtherQueen|Queens], Queen, Offset):-
Queen=\= OtherQueen + Offset,
Queen=\= OtherQueen - Offset,
NewOffset is Offset + 1,
safe(Queens, Queen, NewOffset).
```

Output:

```
queen(8,Solution).

Solution = [1, 3, 5, 7, 2, 4, 6]
Solution = [1, 4, 7, 3, 6, 2, 5]
Solution = [1, 5, 2, 6, 3, 7, 4]
Solution = [1, 6, 4, 2, 7, 5, 3]
Solution = [2, 4, 1, 7, 5, 3, 6]
Solution = [2, 4, 6, 1, 3, 5, 7]
Solution = [2, 5, 1, 4, 7, 3, 6]
Solution = [2, 5, 3, 1, 7, 4, 6]
Solution = [2, 5, 7, 4, 1, 3, 6]
Solution = [2, 6, 3, 7, 4, 1, 5]
```

Program 2 write a program in python to solve N X N queen problem.

Code:

```
result = []
def isSafe(board, row, col):
    for i in range(col):
        if (board[row][i]):
            return False
    i = row
    j = col
    while i >= 0 and j >= 0:
        if(board[i][j]):
            return False
        i -= 1
        j -= 1
    i = row
    j = col
    while j >= 0 and i < n:
        if(board[i][j]):
            return False
        i = i + 1
        j = j - 1
    return True
def solveNQUtil(board, col):
    if (col == n):
        v = []
```

```
            for i in board:
                for j in range(len(i)):
                    if i[j] == 1:
                        v.append(j+1)
            result.append(v)
            return True
        res = False
        for i in range(n):
            if (isSafe(board, i, col)):
                board[i][col] = 1
                res = solveNQUtil(board, col + 1) or res
                board[i][col] = 0
        return res
def solveNQ(n):
        result.clear()
        board = [[0 for j in range(n)]
                for i in range(n)]
        solveNQUtil(board, 0)
        result.sort()
        return result
n = 8
res = solveNQ(n)
for i in range(len(res)):
        print("Solution ",i+1," : ",res[i])
```

output:

```
Solution  1  :  [1, 5, 8, 6, 3, 7, 2, 4]
Solution  2  :  [1, 6, 8, 3, 7, 4, 2, 5]
Solution  3  :  [1, 7, 4, 6, 8, 2, 5, 3]
Solution  4  :  [1, 7, 5, 8, 2, 4, 6, 3]
Solution  5  :  [2, 4, 6, 8, 3, 1, 7, 5]
Solution  6  :  [2, 5, 7, 1, 3, 8, 6, 4]
Solution  7  :  [2, 5, 7, 4, 1, 8, 6, 3]
Solution  8  :  [2, 6, 1, 7, 4, 8, 3, 5]
Solution  9  :  [2, 6, 8, 3, 1, 4, 7, 5]
Solution  10  :  [2, 7, 3, 6, 8, 5, 1, 4]
Solution  11  :  [2, 7, 5, 8, 1, 4, 6, 3]
Solution  12  :  [2, 8, 6, 1, 3, 5, 7, 4]
```