3. The Queens Problem [15 pts]

You are given a standard 8 × 8 chess board. The following problems involve placing queens on the board such that certain constraints are satisfied. For each of the following problems, model the optimization task as an integer program, solve it, and show what an optimal placement of queens on the board looks like.

[1] https://en.wikipedia.org/wiki/Eight_queens_puzzle (https://en.wikipedia.org/wiki/Eight_queens_puzzle)

[2] https://en.wikipedia.org/wiki/Mathematical_chess_problem (https://en.wikipedia.org/wiki/Mathematical_chess_problem)

[3] https://developers.google.com/optimization/puzzles/queens (https://developers.google.com/optimization/puzzles/queens)

[4] https://puzzling.stackexchange.com/questions/22/how-many-chess-pieces-does-it-take-to-cover-all-spaces-on-a-chessboard (https://puzzling.stackexchange.com/questions/22/how-many-chess-pieces-does-it-take-to-cover-all-spaces-on-a-chessboard)

In [2]:

```
# HELPER FUNCTIONS
function printBoard(matrix)
    (height, width) = size(matrix)
    println()

    printBoardLine(width)

    for i in 1:height
        for j in 1:width
            if matrix[i,j] == 1
                print("| X ")
            else
                print("|   ")
            end
        end
        println("|")
        printBoardLine(width)
    end

    println()
end

function printBoardLine(width)

    for w in 1:width
        print("+---")
    end

    println("+")

end
```

Out[2]: printBoardLine (generic function with 1 method)

In [26]:
```
matrix = [ 0 0 0 0 1 1
           0 0 1 0 0 0
           1 0 0 0 0 0
           0 0 0 0 0 1
           0 0 0 1 0 0
           0 1 0 0 0 0 ]

(height, width) = size(matrix)

for i in -5:5
    println(diag(matrix, i))
end
println()

for i in -5:5
    println(diag(flipdim(matrix, 2), i))
end
println()

print("Original board")
printBoard(matrix)

print("Vertically flipped")
printBoard(flipdim(matrix, 1))

print("Horizontally flipped")
printBoard(flipdim(matrix, 2))

print("Rotated 180 ")
printBoard(rot180(matrix))
```

```
[0]
[0,1]
[0,0,0]
[1,0,0,0]
[0,0,0,1,0]
[0,0,0,0,0,0]
[0,1,0,0,0]
[0,0,0,1]
[0,0,0]
[1,0]
[1]

[0]
[0,0]
[1,0,0]
[0,0,1,0]
[0,0,0,0,1]
[1,0,0,0,0,0]
[1,0,0,0,0]
[0,1,0,0]
[0,0,1]
[0,0]
[0]

Original board
+---+---+---+---+---+---+
|   |   |   |   | X | X |
+---+---+---+---+---+---+
|   |   | X |   |   |   |
+---+---+---+---+---+---+
| X |   |   |   |   |   |
+---+---+---+---+---+---+
|   |   |   |   |   | X |
+---+---+---+---+---+---+
|   |   |   | X |   |   |
+---+---+---+---+---+---+
|   | X |   |   |   |   |
+---+---+---+---+---+---+

Vertically flipped
+---+---+---+---+---+---+
|   | X |   |   |   |   |
+---+---+---+---+---+---+
|   |   |   | X |   |   |
+---+---+---+---+---+---+
|   |   |   |   |   | X |
+---+---+---+---+---+---+
| X |   |   |   |   |   |
+---+---+---+---+---+---+
|   |   | X |   |   |   |
+---+---+---+---+---+---+
|   |   |   |   | X | X |
+---+---+---+---+---+---+

Horizontally flipped
+---+---+---+---+---+---+
| X | X |   |   |   |   |
+---+---+---+---+---+---+
|   |   |   | X |   |   |
+---+---+---+---+---+---+
|   |   |   |   |   | X |
+---+---+---+---+---+---+
| X |   |   |   |   |   |
+---+---+---+---+---+---+
|   |   | X |   |   |   |
+---+---+---+---+---+---+
```

a) Find a way to place 8 queens on the board so that no two queens threaten each other. We say that two queens threaten each other if they occupy the same row, column, or diagonal. Show what this placement looks like.

In [32]:
```
# to specify the 8-queen variant of this problem
N = 8

using JuMP, Cbc, Gurobi, Mosek, GLPK

m1 = Model(solver = MosekSolver())

@variable(m1, x[1:N, 1:N], Bin)

# one queen in each row
for i in 1:N
    @constraint(m1, sum(x[i, :]) == 1)
end

# one queen in each column
for j in 1:N
    @constraint(m1, sum(x[:, j]) == 1)
end

# one queen in positive diagonal
for k in -(N-1):(N-1)
    @constraint(m1, sum(diag(x, k)) <= 1)
end

# one queen in negative diagonal
for k in -(N-1):(N-1)
    @constraint(m1, sum(diag(flipdim(x, 2), k)) <= 1)
end

@objective(m1, Max, sum(x))

solve(m1)

printBoard(getvalue(x))
```

```
+---+---+---+---+---+---+---+---+
|   |   | X |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   | X |   |   |
+---+---+---+---+---+---+---+---+
|   | X |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   | X |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   | X |   |   |   |
+---+---+---+---+---+---+---+---+
| X |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   | X |
+---+---+---+---+---+---+---+---+
|   |   |   | X |   |   |   |   |
+---+---+---+---+---+---+---+---+
```

b) Repeat part **(a)** but this time find a placement of the 8 queens that has point symmetry. In other words, find a placement that looks the same if you rotate the board 180 .

In [37]:
```
m2 = Model(solver = MosekSolver())

@variable(m2, x[1:N, 1:N], Bin)

# one queen in each row
for i in 1:N
    @constraint(m2, sum(x[i, :]) == 1)
end

# one queen in each column
for j in 1:N
    @constraint(m2, sum(x[:, j]) == 1)
end

# one queen in positive diagonal
for k in -(N-1):(N-1)
    @constraint(m2, sum(diag(x, k)) <= 1)
end

# one queen in negative diagonal
for k in -(N-1):(N-1)
    @constraint(m2, sum(diag(flipdim(x, 2), k)) <= 1)
end

# point symmetry
@constraint(m2, x .== rot180(x))

@objective(m2, Max, sum(x))

solve(m2)

print("Solution:")
printBoard(getvalue(x))
print("Verification (rotated solution):")
printBoard(rot180(getvalue(x)))
```

```
Solution:
+---+---+---+---+---+---+---+---+
|   |   |   |   |   | X |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   | X |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   | X |   |
+---+---+---+---+---+---+---+---+
| X |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   | X |
+---+---+---+---+---+---+---+---+
|   | X |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   | X |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   | X |   |   |   |   |   |
+---+---+---+---+---+---+---+---+

Verification (rotated solution):
+---+---+---+---+---+---+---+---+
|   |   |   |   |   | X |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   | X |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   | X |   |
+---+---+---+---+---+---+---+---+
| X |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   | X |
+---+---+---+---+---+---+---+---+
|   | X |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   | X |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   | X |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
```

c) What is the smallest number of queens that we can place on the board so that each **empty cell** is threatened by at least one queen? Show a possible optimal placement.

In [45]:
```
m3 = Model(solver = MosekSolver())

@variable(m3, x[1:N, 1:N], Bin)

# one queen in each row
for i in 1:N
    @constraint(m3, sum(x[i, :]) == 1)
end

# one queen in each column
for j in 1:N
    @constraint(m3, sum(x[:, j]) == 1)
end

# one queen in positive diagonal
for k in -(N-1):(N-1)
    @constraint(m3, sum(diag(x, k)) <= 1)
end

# one queen in negative diagonal
for k in -(N-1):(N-1)
    @constraint(m3, sum(diag(flipdim(x, 2), k)) <= 1)
end

# point symmetry
@constraint(m3, x .== rot180(x))

@objective(m3, Min, sum(x))

solve(m3)

print("Solution:")
printBoard(getvalue(x))
print("Verification (rotated solution):")
printBoard(rot180(getvalue(x)))
```

```
Solution:
+---+---+---+---+---+---+---+---+
|   |   |   |   |   | X |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   | X |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   | X |   |
+---+---+---+---+---+---+---+---+
| X |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   | X |
+---+---+---+---+---+---+---+---+
|   | X |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   | X |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   | X |   |   |   |   |   |
+---+---+---+---+---+---+---+---+

Verification (rotated solution):
+---+---+---+---+---+---+---+---+
|   |   |   |   |   | X |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   | X |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   | X |   |
+---+---+---+---+---+---+---+---+
| X |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   | X |
+---+---+---+---+---+---+---+---+
|   | X |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   |   |   | X |   |   |   |
+---+---+---+---+---+---+---+---+
|   |   | X |   |   |   |   |   |
+---+---+---+---+---+---+---+---+
```

d) Repeat part **(c)** but this time find a placement of the queens that also has point symmetry. Does the minimum number of queens required change? Show a possible optimal placement.

In [ ]: