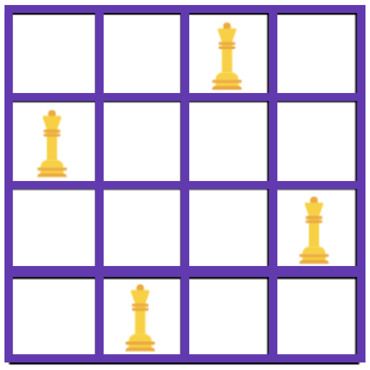
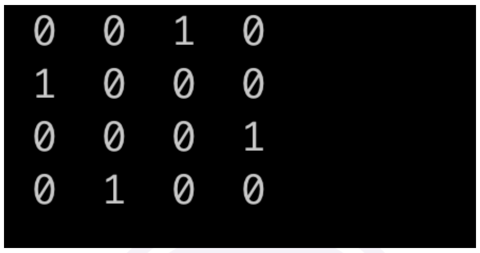
The red cross marks the positions which are under attack from a queen. Whenever we reach a state where we have a queen to place but all the positions in the rows are under attack, we backtrack.

This is not the only possible solution to the problem. If you move each queen one step forward in a clockwise manner, we get another solution.:



Code : LP_CODE2.java

Output.



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

Interview problem: Sudoku Solver

What is Sudoku

Simply put, Sudoku is a combinatorial number placement puzzle with a 9 x 9 grid of cells partially filled with numbers from 1 to 9. The goal is to fill the remaining empty fields with the remaining numbers so that there is only one. Numbers in each row and column. Quantity by type.

Also, each subsection of the 3 x 3 grid cannot have duplicate numbers.
Difficulty increases naturally with the number of empty fields on each board.

For example this is the unsolved Sudoku puzzle

```
8 . . . . . . . .
. . 3 6 . . . . .
. 7 . . 9 . 2 . .
. 5 . . . 7 . . .
. . . . 4 5 7 . .
. . . 1 . . . 3 .
. . 1 . . . . 6 8
. . 8 5 . . . 1 .
. 9 . . . . 4 . .
```

And this is the solved version of it.

```
8 1 2 7 5 3 6 4 9
9 4 3 6 8 2 1 7 5
6 7 5 4 9 1 2 8 3
1 5 4 2 3 7 8 9 6
3 6 9 8 4 5 7 2 1
2 8 7 1 6 9 5 3 4
5 2 1 9 7 4 3 6 8
4 3 8 5 2 6 9 1 7
7 9 6 3 1 8 4 5 2
```

Approach to Solve Sudoku

Sudoku Solving Using Recursive Backtracking Algorithm

As with all other backtracking problems, you can solve Sudoku by sequentially assigning numbers to empty cells.

- Before assigning numbers, we need to check that the current row, current column, and current 3X3 subgrid do not have the same number.
- If there are no numbers in that row, column, or subgrid, you can assign a number and recursively check whether the result of the assignment is a solution. If the operation does not lead to a solution, the next number is tried for the currently empty cell. Returns false if none of the numbers (1 through 9) lead to a solution.
- When we assign a number to a location we need to perform multiple checks on it so that the solution is valid and doesn't break sudoku rules.
- **Row Check:** The following assigned number should not be present in the row
- **Column Check:** The following assigned number should not be present in the column.
- **3X3 box check:** We also need to perform a check the following assigned number shouldn't be present in the enclosing 3X3 box.

To keep the simplicity of code we will check all these constraints in a separate function isValid(). The function will return true, if all the above 3 checks are true else will return false.