Chih-Hsiang Wang, 933271081

// language: Java
// reference: https://www.youtube.com/watch?v=xouin83ebxE
// reference: http://program-lover.blogspot.com/2008/07/eight-queens-puzzle.html
// reference: http://kingxss.iteye.com/blog/2290026

```
package nQueen;
import java.util.HashMap;
import java.util.Map;

public class Queens {
    private Integer queens;
    // check if there is queen in the same column
    private Integer[] column;
    // check if there is queen in the right skew
    private Integer[] rup;
    // check if there is queen in the left skew
    private Integer[] lup;
    // solution
    private Integer[] queen;
    // unique solution
    private Map<String, String> results = new HashMap<String,
String>();
    // index of solution
    private static int idx;
    // num_uniqueber of unique solution
    private static int num_unique = 0;

    // Initialize
    public Queens(int queens) {
        this.queens = queens;
        column = new Integer[queens + 1];
        rup = new Integer[(2 * queens) + 1];
        lup = new Integer[(2 * queens) + 1];
        queen = new Integer[queens + 1];

        for (int i = 0; i <= queens; i++) {
            column[i] = queen[i] = 0;
        }

        for (int i = 0; i <= (2 * queens); i++) {
            rup[i] = lup[i] = 0;
        }
    }

    // The algorithm to solve nQueens problem
    public void backtrack(int i) {
        if (i > queens) {
            showAnswer();
        } else {
            for (int j = 1; j <= queens; j++) {
                if ((column[j] == 0) && (rup[i + j] == 0) && (lup[i -
j + queens] == 0)) {
                    queen[i] = j;
                    column[j] = rup[i + j] = lup[i - j + queens] = 1;
                    backtrack(i + 1);
                    column[j] = rup[i + j] = lup[i - j + queens] = 0;
                }
            }
        }
    }

    // Print out all the unique solutions
    protected void showAnswer() {
        idx++;
        if(!isIndependence(idx)) return;
        System.out.println("solution# " + idx + ":");
        for (int y = 1; y <= queens; y++) {
            for (int x = 1; x <= queens; x++) {
                if (queen[y] == x) {
                    System.out.print("Q");
                } else {
                    System.out.print("x");
                }
            }
            System.out.println(" ");
        }
        System.out.println();
        num_unique += 1;
    }

    // Check if the general solutions independent or not
    protected boolean isIndependence(int idxber) {
        String newSolution = resultToString(queen);
        String flag = results.get(newSolution);
        if (flag != null) {
            return false;
        }

        // symmetric - left & right
        Integer[] leftRight = new Integer[queen.length];
        // symmetric - up & down
        Integer[] upDown = new Integer[queen.length];
        // symmetric - up left - down right
        Integer[] lurd = new Integer[queen.length];
        // symmetric - up right - down left
        Integer[] ruld = new Integer[queen.length];
        // first clockwise rotation
        Integer[] cw1 = new Integer[queen.length];
        for (int i = 1; i < queen.length; i++) {
            leftRight[i] = queen[queen.length - i];
            upDown[i] = queen.length - queen[i];
            lurd[queen.length - queen[i]] = queen.length - i;
            ruld[queen[i]] = i;
            cw1[queen[i]] = queen.length - i;
        }
        // second clockwise rotation
        Integer[] cw2 = new Integer[queen.length];
        for (int i = 1; i < queen.length; i++) {
            cw2[cw1[i]] = queen.length - i;
        }
        // third clockwise rotation
        Integer[] cw3 = new Integer[queen.length];
        for (int i = 1; i < queen.length; i++) {
            cw3[cw2[i]] = queen.length - i;
        }

        results.put(newSolution, idxber + "_self");
        putNewSolution(leftRight, idxber + "_lr");
        putNewSolution(upDown, idxber + "_ud");
        putNewSolution(lurd, idxber + "_lurd");
        putNewSolution(ruld, idxber + "_ruld");
        putNewSolution(cw1, idxber + "_cw1");
        putNewSolution(cw2, idxber + "_cw2");
        putNewSolution(cw3, idxber + "_cw3");

        return true;
    }
```

```
    // Put into hash table to check
    protected void putNewSolution(Integer[] temp, String mark) {
        String newSolution = resultToString(temp);
        String flag = results.get(newSolution);

        if(flag == null) {
            results.put(newSolution, mark);
        }
    }

     // Turn the result into string
    protected String resultToString(Integer[] result) {
        StringBuilder sb = new StringBuilder();
        for (int i = 1; i < queen.length; i++) {
            sb.append(result[i]);
        }
        return sb.toString();
    }

    // Main
    public static void main(String[] args) {
        Queens queen = new Queens(9); // decide the size of board
        queen.backtrack(1);
        System.out.print("Total number of solutions is (including 
refections and rotations): ");
        System.out.print(idx);
        System.out.println();
        System.out.print("Total number of unique solutions is: ");
        System.out.print(num_unique);
    }
}
```

n	total solutions	unique solutions
1	1	1
2	0	0
3	0	0
4	2	1
5	10	2
6	4	1
7	40	6
8	92	12
9	352	46

```
===============================================
When n = 1
Total number of solutions is (including refections and rotations): 1
Total number of unique solutions is: 1

Q

===============================================
When n = 2
Total number of solutions is (including refections and rotations): 0
Total number of unique solutions is: 0

===============================================
When n = 3
Total number of solutions is (including refections and rotations): 0
Total number of unique solutions is: 0

===============================================
When n = 4
Total number of solutions is (including refections and rotations): 2
Total number of unique solutions is: 1

xQxx
xxxQ
Qxxx
xxQx
```

```
===============================================
When n = 5
Total number of solutions is (including refections and rotations): 10
Total number of unique solutions is: 2

Qxxxx
xxQxx
xxxxQ
xQxxx
xxxQx===============================================
When n = 6
Total number of solutions is (including refections and rotations): 4
Total number of unique solutions is: 1

xQxxxx
xxxQxx
xxxxxQ
Qxxxxx
xxQxxx
xxxxQx

===============================================
When n = 7
Total number of solutions is (including refections and rotations): 40
Total number of unique solutions is: 6

Qxxxxxx
xxQxxxx
xxxxQxx
xxxxxxQ
xQxxxxx
xxxQxxx
xxxxxQx

===============================================
When n = 8, there are 92 ways of putting queens.
Total number of solutions is (including refections and rotations): 92
Total number of unique solutions is: 12

Qxxxxxxx
xxxxQxxx
xxxxxxxQ
xxxxxQxx
xxQxxxxx
xxxxxxQx
xQxxxxxx
xxxQxxxx
```

```
================================================
When n = 9
Total number of solutions is (including refections and rotations): 352
Total number of unique solutions is: 46

Qxxxxxxxx
xxQxxxxxx
xxxxxQxxx
xxxxxxxQx
xQxxxxxxx
xxxQxxxxx
xxxxxxxxQ
xxxxxxQxx
xxxxQxxxx
```