/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Online Java Compiler.

Code, Compile, Run and Debug java program online.

Write your code in this editor and press "Run" button to execute it.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

import java.util.\*;

public class Main

{ public static List<List<String>> solveNQueens(int n){

char[][] board =new char[n][n];

for(int i=0;i<n;i++){

for(int j=0;j<n;j++){

board[i][j]='.';

}

}

List<List<String>> res=new ArrayList<List<String>>();

dfs(0,board,res);

return res;

}

static boolean validate(char[][] board,int row,int col){

int duprow=row;

int dupcol=col;

//checking left upper diagonal

while(row>=0&&col>=0){

if(board[row][col]=='Q') return false;

row--;

col--;

}

row=duprow;

col=dupcol;

//checking for left side

while(col>=0){

if(board[row][col]=='Q') return false;

col--;

}

row=duprow;

col=dupcol;

//checking for left lower diagonal

while(col>=0&&row<board.length){

if(board[row][col]=='Q') return false;

col--;

row++;

}

return true;

}

static void dfs(int col,char[][] board,List<List<String>> res){

if(col==board.length){

res.add(construct(board));

return;

}

for(int row=0;row<board.length;row++){

if(validate(board,row,col)){

board[row][col]='Q';

dfs(col+1,board,res);

board[row][col]='.';

}

}

}

static List<String> construct(char[][] board){

List<String> res=new LinkedList<String>();

for(int i=0;i<board.length;i++){

String s=new String(board[i]);

res.add(s);

}

return res;

}

public static void main(String[] args) {

int N=9;

List<List<String>> queen=solveNQueens(N);

int i=1;

int count=queen.size();

for(List<String> it:queen){

System.out.println("Arrangement"+i);

for(String s:it){

System.out.println(s);

}

System.out.println();

i+=1;

}

System.out.println(count);

}

}