//Assignment 15

//1.

**package** psr;

**import** java.util.\*;

**import** java.util.\*;

**public** **class** homework {

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

List<List<String>> results = **new** ArrayList<>();

**int**[] board = **new** **int**[n];

backtrack(results, board, 0, n);

**return** results;

}

**private** **void** backtrack(List<List<String>> results, **int**[] board, **int** row, **int** n) {

**if** (row == n) {

results.add(createBoard(board, n));

**return**;

}

**for** (**int** col = 0; col < n; col++) {

**if** (isValid(board, row, col)) {

board[row] = col;

backtrack(results, board, row + 1, n);

}

}

}

**private** **boolean** isValid(**int**[] board, **int** row, **int** col) {

**for** (**int** i = 0; i < row; i++) {

**if** (board[i] == col || Math.*abs*(board[i] - col) == row - i)

**return** **false**;

}

**return** **true**;

}

**private** List<String> createBoard(**int**[] board, **int** n) {

List<String> boardStr = **new** ArrayList<>();

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

**char**[] row = **new** **char**[n];

Arrays.*fill*(row, '.');

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

boardStr.add(**new** String(row));

}

**return** boardStr;

}

**public** **static** **void** main(String[] args) {

homework solver = **new** homework();

List<List<String>> solutions = solver.solveNQueens(4);

**for** (List<String> sol : solutions) {

**for** (String row : sol) {

System.***out***.println(row);

}

System.***out***.println();

}

}

}

//2.

**package** psr;

**import** java.util.\*;

**public** **class** homework {

**private** **int** n;

**private** **int**[][] maze;

**private** List<String> results = **new** ArrayList<>();

**private** **boolean**[][] visited;

**public** List<String> findPaths(**int**[][] maze) {

**this**.n = maze.length;

**this**.maze = maze;

**this**.visited = **new** **boolean**[n][n];

**if** (maze[0][0] == 1)

backtrack(0, 0, **new** StringBuilder());

**return** results;

}

**private** **void** backtrack(**int** x, **int** y, StringBuilder path) {

**if** (x == n - 1 && y == n - 1) {

results.add(path.toString());

**return**;

}

visited[x][y] = **true**;

**int**[] dx = {1, 0, 0, -1};

**int**[] dy = {0, -1, 1, 0};

**char**[] dir = {'D', 'L', 'R', 'U'};

**for** (**int** i = 0; i < 4; i++) {

**int** nx = x + dx[i];

**int** ny = y + dy[i];

**if** (isSafe(nx, ny)) {

path.append(dir[i]);

backtrack(nx, ny, path);

path.deleteCharAt(path.length() - 1);

}

}

visited[x][y] = **false**;

}

**private** **boolean** isSafe(**int** x, **int** y) {

**return** x >= 0 && x < n &&

y >= 0 && y < n &&

maze[x][y] == 1 &&

!visited[x][y];

}

**public** **static** **void** main(String[] args) {

**int**[][] maze = {

{1, 0, 0, 0},

{1, 1, 0, 1},

{0, 1, 0, 0},

{1, 1, 1, 1}

};

homework solver = **new** homework();

List<String> paths = solver.findPaths(maze);

System.***out***.println(paths);

}

}

//3.

**package** psr;

**import** java.util.\*;

**public** **class** homework {

**private** **static** **final** String[] ***mapping*** = {

"",

"",

"abc",

"def",

"ghi",

"jkl",

"mno",

"pqrs",

"tuv",

"wxyz"

};

**public** List<String> letterCombinations(String digits) {

List<String> results = **new** ArrayList<>();

**if** (digits == **null** || digits.length() == 0) **return** results;

backtrack(digits, 0, **new** StringBuilder(), results);

**return** results;

}

**private** **void** backtrack(String digits, **int** index, StringBuilder current, List<String> results) {

**if** (index == digits.length()) {

results.add(current.toString());

**return**;

}

**int** digit = digits.charAt(index) - '0';

String letters = ***mapping***[digit];

**for** (**char** c : letters.toCharArray()) {

current.append(c);

backtrack(digits, index + 1, current, results);

current.deleteCharAt(current.length() - 1);

}

}

**public** **static** **void** main(String[] args) {

homework solver = **new** homework();

System.***out***.println(solver.letterCombinations("23"));

}

}