**Assignment 15**

**1.NQueens**

import java.util.\*;

public class NQueens {

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

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

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

for (char[] row : board)

Arrays.fill(row, '.');

solve(0, board, result, n);

return result;

}

private static void solve(int row, char[][] board, List<List<String>> result, int n) {

if (row == n) {

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

for (char[] chars : board)

list.add(new String(chars));

result.add(list);

return;

}

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

if (isSafe(board, row, col, n)) {

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

solve(row + 1, board, result, n);

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

}

}

private static boolean isSafe(char[][] board, int row, int col, int n) {

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

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

for (int i = row - 1, j = col - 1; i >= 0 && j >= 0; i--, j--)

if (board[i][j] == 'Q') return false;

for (int i = row - 1, j = col + 1; i >= 0 && j < n; i--, j++)

if (board[i][j] == 'Q') return false;

return true;

}

public static void main(String[] args) {

int n = 4;

List<List<String>> solutions = solveNQueens(n);

for (List<String> sol : solutions) {

for (String row : sol)

System.out.println(row);

System.out.println();

}

}

}

**2.Rat in maze**

import java.util.\*;

public class RatMaze {

static List<String> paths;

static int[] dx = {1, 0, 0, -1}; // D, L, R, U

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

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

public static List<String> findPaths(int[][] maze, int n) {

paths = new ArrayList<>();

if (maze[0][0] == 0 || maze[n - 1][n - 1] == 0)

return paths;

boolean[][] visited = new boolean[n][n];

dfs(maze, 0, 0, "", visited, n);

return paths;

}

private static void dfs(int[][] maze, int x, int y, String path, boolean[][] visited, int n) {

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

paths.add(path);

return;

}

visited[x][y] = true;

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

int newX = x + dx[i], newY = y + dy[i];

if (newX >= 0 && newY >= 0 && newX < n && newY < n &&

maze[newX][newY] == 1 && !visited[newX][newY]) {

dfs(maze, newX, newY, path + dir[i], visited, n);

}

}

visited[x][y] = false;

}

public static void main(String[] args) {

int[][] maze = {

{1, 0, 0, 0},

{1, 1, 0, 1},

{0, 1, 0, 0},

{1, 1, 1, 1}

};

List<String> result = findPaths(maze, 4);

System.out.println(result);

}

}

**3. Letter combination of phone number**

import java.util.\*;

public class PhoneLetterCombinations {

public List<String> letterCombinations(String digits) {

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

if (digits.isEmpty())

return result;

String[] map = {

"abc", "def", "ghi",

"jkl", "mno", "pqrs", "tuv", "wxyz"

};

backtrack(result, digits, new StringBuilder(), 0, map);

return result;

}

private void backtrack(List<String> result, String digits, StringBuilder current, int index, String[] map) {

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

result.add(current.toString());

return;

}

String letters = map[digits.charAt(index) - '0'];

for (char ch : letters.toCharArray()) {

current.append(ch);

backtrack(result, digits, current, index + 1, map);

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

}

}

public static void main(String[] args) {

PhoneLetterCombinations plc = new PhoneLetterCombinations();

String input = "23";

List<String> output = plc.letterCombinations(input);

System.out.println(output);

}

}