

Code : 1

```
package Topic_11_RecursionBacktracking;
```

```
import java.util.Scanner;
```

```
public class A_FloodFill {
```

```
    public static void main(String[] args) throws Exception {
```

```
        Scanner scn = new Scanner(System.in);
```

```
        int n = scn.nextInt();
```

```
        int m = scn.nextInt();
```

```
        int[][] arr = new int[n][m];
```

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

```
            for (int j = 0; j < m; j++) {
```

```
                arr[i][j] = scn.nextInt();
```

```
            }
```

```
        }
```

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

```
        floodfill(arr, 0, 0, "", visited);
```

```
    }
```

```
    public static void floodfill(int[][] maze, int sr, int sc, String asf, boolean[][] visited) {
```

```
        if (sr < 0 || sc < 0 || sr == maze.length || sc == maze[0].length || maze[sr][sc] == 1  
            || visited[sr][sc] == true) {
```

```
            return;
```

```
        }
```

```
        if (sr == maze.length - 1 && sc == maze[0].length - 1) {
```

```
            System.out.println(asf);
```

```
            return;
```

```
        }
```

```
        visited[sr][sc] = true;
```

```
        floodfill(maze, sr - 1, sc, asf + "t", visited);
```

```
        floodfill(maze, sr, sc - 1, asf + "l", visited);
```

```
        floodfill(maze, sr + 1, sc, asf + "d", visited);
```

```
        floodfill(maze, sr, sc + 1, asf + "r", visited);
```

```
    }
```

```
}
```

Code : 2

```
package Topic_11_RecursionBacktracking;
```

```
import java.io.*;
```

```
import java.util.*;
```

```
public class B_TargetSumSubset {
```

```
    public static void main(String[] args) throws Exception {
```

```
        Scanner scn = new Scanner(System.in);
```

```
        int n = scn.nextInt();
```

```
        int[] arr = new int[n];
```

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

```
            arr[i] = scn.nextInt();
```

```
        }
```

```
        int tar = scn.nextInt();
```

```
        printTargetSumSubsets(arr, 0, "", 0, tar); //1
```

```
    }
```

```
    // set is the subset
```

```
    // sos is sum of subset
```

```
    // tar is target
```

```
    public static void printTargetSumSubsets(int[] arr, int idx, String set, int sos, int tar) {
```

```
        if (idx == arr.length) { //2
```

```
            if (sos == tar) {
```

```
                System.out.println(set + ".");
```

```
            }
```

```
            return;
```

```
        }
```

```
        printTargetSumSubsets(arr, idx + 1, set + arr[idx] + ", ", sos + arr[idx], tar); //3
```

```
        printTargetSumSubsets(arr, idx + 1, set, sos, tar); //4
```

```
    }
```

```
}
```

Code : 3

```
package Topic_11_RecursionBacktracking;
```

```
import java.io.*;
```

```
import java.util.*;
```

```
public class C_NQueenProblem {
```

```
    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        int n = Integer.parseInt(br.readLine());
        int[][] chess = new int[n][n];
        printNQueens(chess, "", 0);
    }
```

```
    public static void printNQueens(int[][] chess, String qsf, int row) {
        if (row == chess.length) {
            System.out.println(qsf + ".");
            return;
        }
        for (int col = 0; col < chess.length; col++) {
            if (chess[row][col] == 0 && isQueenSafe(chess, row, col) == true) {
                chess[row][col] = 1;
                printNQueens(chess, qsf + row + "-" + col + " ", row + 1);
                chess[row][col] = 0;
            }
        }
    }
```

```
    public static boolean isQueenSafe(int[][] chess, int row, int col) {
        for (int i = row - 1, j = col - 1; i >= 0 && j >= 0; i--, j--) {
            if (chess[i][j] == 1) {
                return false;
            }
        }

        for (int i = row - 1, j = col; i >= 0; i--) {
            if (chess[i][j] == 1) {
                return false;
            }
        }

        for (int i = row - 1, j = col + 1; i >= 0 && j < chess.length; i--, j++) {
            if (chess[i][j] == 1) {
                return false;
            }
        }

        for (int i = row, j = col - 1; j >= 0; j--) {
            if (chess[i][j] == 1) {
                return false;
            }
        }

        return true;
    }
```

} }

Code : 4

```
package Topic_11_RecursionBacktracking;
```

```
import java.io.*;
```

```
import java.util.*;
```

```
public class D_KnightsTour {
```

```
    public static void main(String[] args) throws Exception {
```

```
        Scanner s = new Scanner(System.in);
```

```
        int n = s.nextInt();
```

```
        int r = s.nextInt();
```

```
        int c = s.nextInt();
```

```
        int[][] chess = new int[n][n];
```

```
        printKnightsTour(chess, r, c, 1);
```

```
    }
```

```
    private static void printKnightsTour(int[][] chess, int r, int c, int move) {
```

```
        // TODO Auto-generated method stub
```

```
        if (r < 0 || c < 0 || r >= chess.length || c >= chess.length || chess[r][c] > 0) {
```

```
            return;
```

```
        } else if (move == chess.length * chess.length) {
```

```
            chess[r][c] = move;
```

```
            displayBoard(chess);
```

```
            chess[r][c] = 0;
```

```
            return;
```

```
        }
```

```
        chess[r][c] = move;
```

```
        printKnightsTour(chess, r - 2, c + 1, move + 1);
```

```
        printKnightsTour(chess, r - 1, c + 2, move + 1);
```

```
        printKnightsTour(chess, r + 1, c + 2, move + 1);
```

```
        printKnightsTour(chess, r + 2, c + 1, move + 1);
```

```
        printKnightsTour(chess, r + 2, c - 1, move + 1);
```

```
        printKnightsTour(chess, r + 1, c - 2, move + 1);
```

```
        printKnightsTour(chess, r - 1, c - 2, move + 1);
```

```
        printKnightsTour(chess, r - 2, c - 1, move + 1);
```

```
        chess[r][c] = 0;
```

```
    }
```

```
    private static void displayBoard(int[][] a) {
```

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

```
            for (int j = 0; j < a[0].length; j++) {
```

```
                System.out.print(a[i][j] + " ");
```

```
            }
```

```
            System.out.println();
```

```
        }
```

```
        System.out.println();
```

```
    }
```

```
}
```

