Name Of the Student : Omkar Arun Shinde

Class / Division : TY – D

PRN No. : 122B1B258

Roll No. : 122B1B258

**Assignment No. 05**

**Problem Statement** : A classic problem that can be solved by backtracking is called the Knight's tour Problem. It is a problem in which we are provided with a NxN chessboard and a knight. For a person who is not familiar with chess, the knight moves two squares horizontally and one square vertically, or two squares vertically and one square horizontally. In this problem, there is an empty chess board, and a knight starting from any location in the board, our task is to check whether the knight can visit all of the squares in the board or not. When It can visit all of the squares, then place the number of jumps needed to reach that location from the starting point.

**Code**

package Lab\_08Oct;

import java.util.\*;

class Knight\_Tour {

int N;

Knight\_Tour() {}

Knight\_Tour(int N) {

this.N = N;

}

public void Solution\_Knight\_Tour(int x, int y) {

int Chess[][] = new int[N][N];

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

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

Chess[i][j] = -1;

}

}

int MovX[] = { 2, 1, -1, -2, -2, -1, 1, 2 };

int MovY[] = { 1, 2, 2, 1, -1, -2, -2, -1 };

Chess[x][y] = 0;

if (!Tour(x, y, 1, Chess, MovX, MovY)) {

System.*out*.println("Solution does not exist for starting position.");

return;

}

else {

System.*out*.println("Solution does exist for starting position.");

PrintChessBoard(Chess);

}

}

public boolean Tour(int x, int y, int movei, int Chess[][], int MovX[], int MovY[]) {

int k, x\_new, y\_new;

if (movei == N \* N)

return true;

for (k = 0; k < 8; k++) {

x\_new = x + MovX[k];

y\_new = y + MovY[k];

if (Safe(x\_new, y\_new, Chess)) {

Chess[x\_new][y\_new] = movei;

if (Tour(x\_new, y\_new, movei + 1, Chess, MovX, MovY))

return true;

else {

Chess[x\_new][y\_new] = -1;

}

}

}

return false;

}

public boolean Safe(int x, int y, int Chess[][])

{

return (x >= 0 && x < N && y >= 0 && y < N && Chess[x][y] == -1);

}

public void PrintChessBoard(int Chess[][]) {

for (int x = 0; x < N; x++) {

for (int y = 0; y < N; y++) {

System.*out*.print(Chess[x][y] + " ");

}

System.*out*.println();

}

}

}

public class Knight\_Tour\_Problem {

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner sc = new Scanner(System.*in*);

int n;

System.*out*.print("Enter number of one dimension of chess: ");

n = sc.nextInt();

int x, y;

System.*out*.print("Enter starting X position in Chess board: ");

x = sc.nextInt();

System.*out*.print("Enter starting Y position in Chess board: ");

y = sc.nextInt();

Knight\_Tour K = new Knight\_Tour(n);

K.Solution\_Knight\_Tour(x, y);

}

}

**Output**

Enter number of one dimension of chess: 8

Enter starting X position in Chess board: 0

Enter starting Y position in Chess board: 0

Solution does exist for starting position.

0 59 38 33 30 17 8 63

37 34 31 60 9 62 29 16

58 1 36 39 32 27 18 7

35 48 41 26 61 10 15 28

42 57 2 49 40 23 6 19

47 50 45 54 25 20 11 14

56 43 52 3 22 13 24 5

51 46 55 44 53 4 21 12

Enter number of one dimension of chess: 4

Enter starting X position in Chess board: 0

Enter starting Y position in Chess board: 0

Solution does not exist for starting position.