using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace KnightTourpro

{

class KnightTour

{

public const int Row = 8;

public const int Col = 8;

public char GridSymbol = '#';

public char MoveSymbol = 'X';

public int[] startPosition = new int[2];

public char[,] moves = new char[Row, Col];

public int[] PossibleMovesX = new int[] { -1, 1, -2, 2, 1, -1, 2, -2 };

public int[] PossibleMovesY = new int[] { 2, 2, 1, 1, -2, -2, -1, -1 };

public int hMove;

public int vMove;

public int roundMoves = 0;

public int bestMoves = 0;

public void MainFunction()

{

bool isOver = false;

Random rand = new Random();

startPosition[0] = rand.Next(0, Row);

startPosition[1] = rand.Next(0, Col);

moves[startPosition[0], startPosition[1]] = MoveSymbol;

bool gameOver = false;

int counter = 0;

int bestRoundCounter = 0;

while (!gameOver && counter < 100000)

{

PopulateArray();

isOver = false;

while (!isOver)

{

isOver = MakeMove();

}

counter++;

gameOver = areBlocksFilled();

if (gameOver)

{

DisplayGrid();

Console.WriteLine("It took {0} tries to win the game", counter);

}

else if (roundMoves > bestMoves)

{

bestRoundCounter = counter;

bestMoves = roundMoves;

DisplayGrid();

}

}

}

public bool areBlocksFilled()

{

roundMoves = 0;

bool blocksAreaFilled = true;

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

{

for (int c = 0; c < Col; c++)

{

if (moves[i, c] != GridSymbol)

roundMoves++;

else

{

blocksAreaFilled = false;

}

}

}

return blocksAreaFilled;

}

public bool MakeMove()

{

for (int i = 0; i < 8; i++)

{

if (!getMove())

{

return true;

}

}

return false;

}

public bool getMove()

{

Random rand = new Random();

int badMoveCounter = 0;

int[] badMoves = new int[] { -1, -1, -1, -1, -1, -1, -1, -1 };

while (badMoveCounter < 8)

{

int randomMove = rand.Next(0, 8);

hMove = startPosition[0] + PossibleMovesX[randomMove];

vMove = startPosition[1] + PossibleMovesY[randomMove];

if (((hMove < Row && hMove >= 0) && (vMove < Col && vMove >= 0))&&

(moves[hMove, vMove] != MoveSymbol))

{

moves[hMove, vMove] = MoveSymbol;

startPosition[0] = hMove;

startPosition[1] = vMove;

return true;

}

else

{

bool moveAlreadyMade = false;

for (int i = 0; i < 8; i++)

{

if (randomMove == badMoves[i])

{

moveAlreadyMade = true;

}

}

if (!moveAlreadyMade)

{

badMoves[badMoveCounter] = randomMove;

badMoveCounter++;

}

}

}

return false;

}

public void DisplayGrid()

{

Console.Write(" ");

for (int i = 65; i < Col + 65; i++)

Console.Write(Convert.ToChar(i));

Console.WriteLine();

for (int r = 0; r < Row; r++)

{

Console.Write(r + 1);

for (int c = 0; c < Col; c++)

{

Console.Write(moves[r, c]);

}

Console.WriteLine();

}

}

public void PopulateArray()

{

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

{

for (int c = 0; c < Col; c++)

{

moves[i, c] = GridSymbol;

}

}

}

}

class Program

{

static void Main(string[] args)

{

Console.WriteLine("\t\tKnight Tour Problem");

KnightTour KT = new KnightTour();

KT.MainFunction();

}

}

}