using System;

using System.Diagnostics.Metrics;

internal class Program

{

private static void Main()

{

int counts1 = 0;

int counts2 = 0;

int counts10 = 0;

int counts11 = 0;

int counts4 = 0;

int counts3 = 0;

int counts5 = 0;

int counts6 = 0;

int counts7 = 0;

int counts8 = 0;

int counts9 = 0;

int counts12 = 0;

int counts13 = 0;

int counts14 = 0;

int[,] field = new int[5, 5];

int playerScore = 0;

int computerScore = 0;

Random random = new();

while (!IsFieldFull(field))

{

int number = random.Next(1, 15);

int row, column;

// Ход игрока

Console.WriteLine("Введите координаты клетки (например, 1 2):");

string[] input = Console.ReadLine().Split();

row = int.Parse(input[0]) - 1;

column = int.Parse(input[1]) - 1;

field[row, column] = number;

playerScore += CalculatePoints(GetNumbersInRow(field, row), counts1, counts2, counts10, counts11, counts4, counts3, counts5, counts6, counts7, counts8, counts9, counts12, counts13, counts14) + CalculatePoints(GetNumbersInColumn(field, column), counts1, counts2, counts10, counts11, counts4, counts3, counts5, counts6, counts7, counts8, counts9, counts12, counts13, counts14) + CalculatePoints(GetNumbersInDiagonal(field), counts1, counts2, counts10, counts11, counts4, counts3, counts5, counts6, counts7, counts8, counts9, counts12, counts13, counts14);

// Ход компьютера

int bestRow = -1, bestColumn = -1, bestPoints = -1;

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

if (field[i, j] == 0)

{

field[i, j] = number;

int points = CalculatePoints(GetNumbersInRow(field, i), counts1, counts2, counts10, counts11, counts4, counts3, counts5, counts6, counts7, counts8, counts9, counts12, counts13, counts14) + CalculatePoints(GetNumbersInColumn(field, j), counts1, counts2, counts10, counts11, counts4, counts3, counts5, counts6, counts7, counts8, counts9, counts12, counts13, counts14) + CalculatePoints(GetNumbersInDiagonal(field), counts1, counts2, counts10, counts11, counts4, counts3, counts5, counts6, counts7, counts8, counts9, counts12, counts13, counts14);

if (points > bestPoints)

{

bestRow = i;

bestColumn = j;

bestPoints = points;

}

field[i, j] = 0;

}

}

}

field[bestRow, bestColumn] = number;

computerScore += bestPoints;

PrintField(field);

}

Console.WriteLine("Игра окончена.");

Console.WriteLine($"Очки игрока: {playerScore}");

Console.WriteLine($"Очки компьютера: {computerScore}");

bool IsFieldFull(int[,] field)

{

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

if (field[i, j] == 0)

{

return false;

}

}

}

return true;

}

int[] GetNumbersInRow(int[,] field, int row)

{

int[] numbers = new int[5];

for (int i = 0; i < 5; i++)

{

numbers[i] = field[row, i];

}

return numbers;

}

int[] GetNumbersInColumn(int[,] field, int column)

{

int[] numbers = new int[5];

for (int i = 0; i < 5; i++)

{

numbers[i] = field[i, column];

}

return numbers;

}

int[] GetNumbersInDiagonal(int[,] field)

{

int[] numbers = new int[5];

for (int i = 0; i < 5; i++)

{

numbers[i] = field[i, i];

}

return numbers;

}

}

private static void PrintField(int[,] field)

{

throw new NotImplementedException();

}

private static int CalculatePoints(int[] numbers, int counts1, int counts2, int counts10, int counts11, int counts4, int counts3, int counts5, int counts6, int counts7, int counts8, int counts9, int counts12, int counts13, int counts14)

{

Array.Sort(numbers);

int count = 0;

for (int i = 0; i < 5; i++)

{

if (numbers[i] == 0)

{

count++;

}

}

if (count == 5)

{

return 0;

}

if (numbers[0] == numbers[4])

{

return 160;

}

if (IsConsecutive(numbers))

{

return 50;

}

int[] countsnumbers = new int[15];

for (int i = 0; i < 5; i++)

{

countsnumbers[i]++;

}

if (counts1 == 3 && counts2 == 2)

{

return 100;

}

if (counts1 == 1 && counts10 == 1 && counts11 == 1 && counts12 == 1 && counts13 == 1)

{

return 150;

}

if (counts1 == 4 || counts2 == 4 || counts3 == 4 || counts4 == 4 || counts5 == 4 || counts6 == 4 || counts7 == 4 || counts8 == 4 || counts9 == 4 || counts10 == 4 || counts11 == 4 || counts12 == 4 || counts13 == 4 || counts14 == 4)

{

return 40;

}

if (counts1 == 2 && counts2 == 2 || counts1 == 2 && counts3 == 2 || counts1 == 2 && counts4 == 2 || counts1 == 2 && counts5 == 2 || counts1 == 2 && counts6 == 2 || counts1 == 2 && counts7 == 2 || counts1 == 2 && counts8 == 2 || counts1 == 2 && counts9 == 2 || counts1 == 2 && counts10 == 2 || counts1 == 2 && counts11 == 2 || counts1 == 2 && counts12 == 2 || counts1 == 2 && counts13 == 2 || counts1 == 2 && counts14 == 2 || counts2 == 2 && counts3 == 2 || counts2 == 2 && counts4 == 2 || counts2 == 2 && counts5 == 2 || counts2 == 2 && counts6 == 2 || counts2 == 2 && counts7 == 2 || counts2 == 2 && counts8 == 2 || counts2 == 2 && counts9 == 2 || counts2 == 2 && counts10 == 2 || counts2 == 2 && counts11 == 2 || counts2 == 2 && counts12 == 2 || counts2 == 2 && counts13 == 2 || counts2 == 2 && counts14 == 2 || counts3 == 2 && counts4 == 2 || counts3 == 2 && counts5 == 2 || counts3 == 2 && counts6 == 2 || counts3 == 2 && counts7 == 2 || counts3 == 2 && counts8 == 2 || counts3 == 2 && counts9 == 2 || counts3 == 2 && counts10 == 2 || counts3 == 2 && counts11 == 2 || counts3 == 2 && counts12 == 2 || counts3 == 2 && counts13 == 2 || counts3 == 2 && counts14 == 2 || counts4 == 2 && counts5 == 2 || counts4 == 2 && counts6 == 2 || counts4 == 2 && counts7 == 2 || counts4 == 2 && counts8 == 2 || counts4 == 2 && counts9 == 2 || counts4 == 2 && counts10 == 2 || counts4 == 2 && counts11 == 2 || counts4 == 2 && counts12 == 2 || counts4 == 2 && counts13 == 2 || counts4 == 2 && counts14 == 2 || counts5 == 2 && counts6 == 2 || counts5 == 2 && counts7 == 2 || counts5 == 2 && counts8 == 2 || counts5 == 2 && counts9 == 2 || counts5 == 2 && counts10 == 2 || counts5 == 2 && counts11 == 2 || counts5 == 2 && counts12 == 2 || counts5 == 2 && counts13 == 2 || counts5 == 2 && counts14 == 2 || counts6 == 2 && counts7 == 2 || counts6 == 2 && counts8 == 2 || counts6 == 2 && counts9 == 2 || counts6 == 2 && counts10 == 2 || counts6 == 2 && counts11 == 2 || counts6 == 2 && counts12 == 2 || counts6 == 2 && counts13 == 2 || counts6 == 2 && counts14 == 2 || counts7 == 2)

{

return 111;

}

}

private static bool IsConsecutive(int[] numbers)

{

throw new NotImplementedException();

}

}