Proficiency Profile™

IKM TeckChek

Naveen Seerapu

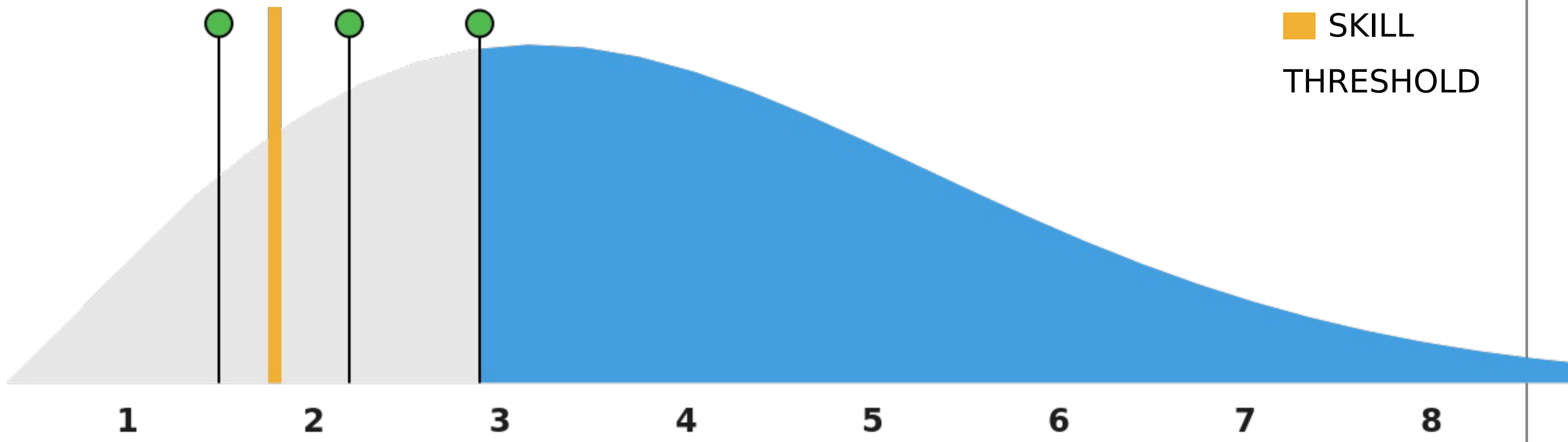
E-MAIL
naveen97869@gmail.com

DATE OF LAST ACTIVITY
2021-02-14

TEST
C# Code Challenge - Basic

LANGUAGES USED
C#

Technical Skill



🟦 CANDIDATE RESULT
🟧 SKILL THRESHOLD

●	Strongly recommended regarding technical potential.
○	Recommended regarding technical potential.
○	More information needed.
○	Not recommended if technical ability is crucial.

TASKS IN TEST	CORRECT	SUBMISSIONS	PLAGIARISM	LANGUAGE

EASY **Dice Game**	✓	3	✓	C#
EASY **Radio Commercials**	✓	1	✓	C#
MEDIUM **Chess**	✓	3	✓	C#

EASY

Dice Game

SHOWN AS	MEMORY LIMIT	CORRECTNESS	PLAGIARISM
Problem A	**1024 MB**	✓ **Accepted**	✓ **Not detected**
CPU TIME LIMIT	LANGUAGE USED	ATTEMPTS	EXPLANATIONS
1 second	**C#**	**3**	✓ **Added**

Gunnar and Emma play a lot of board games at home, so they own many dice that are not normal 6-sided dice. For example they own a die that has 10 sides with numbers $47, 48, \ldots, 56$ on it.

Photo by JD Hancock (https://www.flickr.com/photos/jdhanc

There has been a big storm in Stockholm, so Gunnar and Emma have been stuck at home without electricity for a couple of hours. They have finished playing all the games they have, so they came up with a new one. Each player has 2 dice which he or she rolls. The player with a bigger sum wins. If both sums are the same, the game ends in a tie.

Task

Given the description of Gunnar's and Emma's dice, which player has higher chances of winning?

All of their dice have the following property: each die contains numbers $a, a+1, \ldots, b$, where a and b are the lowest and highest numbers respectively on the die. Each number appears exactly on one side, so the die has $b-a+1$ sides.

Input

The first line contains four integers a_1, b_1, a_2, b_2 that describe Gunnar's dice. Die number i contains numbers $a_i, a_i + 1, \ldots, b_i$ on its sides. You may assume that $1 \leq a_i \leq b_i \leq 100$. You can further assume that each die has at least four sides, so $a_i + 3 \leq b_i$.

The second line contains the description of Emma's dice in the same format.

Output

Output the name of the player that has higher probability of winning. Output "Tie" if both players have same probability of winning.

Sample Input 1

```
1 4 1 4
1 6 1 6
```

Sample Output 1

```
Emma
```

Sample Input 2

```
1 8 1 8
1 10 2 5
```

Sample Output 2

```
Tie
```

Sample Input 3

```
2 5 2 7
1 5 2 5
```

Sample Output 3

```
Gunnar
```

e74f2a5161dd5498.cs

```
using System;

namespace DiceProblem
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                long gunnar_dice1_start, gunnar_dice1_end, gunnar_dice2_start, gunnar_dice2_end,
                    emma_dice1_start, emma_dice1_end, emma_dice2_start, emma_dice2_end;
                double meanGunnar, meanEmma;
                int[] gunnarArr = Array.ConvertAll(Console.ReadLine().Split(" "), i => int.Parse(i));
                int[] emmaArr = Array.ConvertAll(Console.ReadLine().Split(" "), i => int.Parse(i));

                gunnar_dice1_start = gunnarArr[0];
                gunnar_dice1_end = gunnarArr[1];
                gunnar_dice2_start = gunnarArr[2];
                gunnar_dice2_end = gunnarArr[3];

                emma_dice1_start = emmaArr[0];
                emma_dice1_end = emmaArr[1];
                emma_dice2_start = emmaArr[2];
                emma_dice2_end = emmaArr[3];

                meanGunnar = (gunnar_dice1_start + gunnar_dice1_end) / 2.0 + (gunnar_dice2_start + gunnar_dice2_end) / 2.0;
                meanEmma = (emma_dice1_start + emma_dice1_end) / 2.0 + (emma_dice2_start + emma_dice2_end) / 2.0;
                if (meanGunnar > meanEmma)
                    Console.WriteLine("Gunnar");
                else if (meanGunnar < meanEmma)
                    Console.WriteLine("Emma");
                else
                    Console.WriteLine("Tie");

            }
            catch(Exception e)
            {
                Environment.Exit(0);
            }
            finally
            {
                Environment.Exit(0);
            }

        }


    }
}

```

Explanations provided by the candidate

Explanation of the problem: No explanation given

Explanation of the Solution: Considering the average of the smallest and largest values on the face of Dice can be used to compare the winning probability in this scenario.

Proficiency Profile™

IKM TeckChek

EASY

Radio Commercials

SHOWN AS
Problem B

CPU TIME LIMIT
1 second

MEMORY LIMIT
1024 MB

LANGUAGE USED
C#

CORRECTNESS
✓ **Accepted**

ATTEMPTS
1

PLAGIARISM
✓ **Not detected**

EXPLANATIONS
✓ **Added**

Our favorite Onid Pizza would like to have a commercial aired in a radio. Since they are close to KTH, they want to attract mainly students. It's not a good idea to have the commercial aired between 8 am and 5 pm, because most of the students are in the school and don't listen to the radio. Onid made a survey and now they know how many students listen to the radio at each point of the day.

They also know that if each student listens to a commercial, he or she will spend one Swedish crown on pizza in expectation. Thus if 100 students listen to a commercial, Onid will increase their income by 100 crowns on average from selling more pizza.

Of course, Onid Pizza has to pay a fixed amount every time the commercial is played. The radio has a commercial break every 15 minutes. Unfortunately, Onid can choose only one continuous sequence of commercial breaks, for example all breaks from 5 pm to 8 pm. Help them to choose a continuous sequence of commercial breaks such that their profit is maximal.

Input

The first line of the input contains two space separated positive integers N, P – the total number of commercial breaks in a day and the price of one commercial break. You can assume that $N \leq 100\,000$ and $P \leq 1\,000$. On the next line there are N space-separated integers – the i'th integer denotes how many students listen to the i-th commercial break. There are always at most 2 000 students listening.

Output

Output contains one line with one integer – the biggest expected extra profit Onid can get by selecting a continuous sequence of commercial breaks.

Sample Input 1

```
6 20
18 35 6 80 15 21
```

Sample Output 1

```
61
```

28960591009d6bd3.cs

```
using System;

namespace OnidPizza
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                int N, P; bool isValidInput = false;
                int[] strNandP = Array.ConvertAll(Console.ReadLine().Split(" "), i => int.Parse(i));

                N = strNandP[0];
                P = strNandP[1];

                isValidInput = (N >= 0) && (P >= 0);
                if (!isValidInput)
                    Environment.Exit(0);

                string strInput = Console.ReadLine();
                isValidInput = strInput.Trim().Length > 0;
                if (!isValidInput)
                    Environment.Exit(0);

                string[] strSeq = strInput.Split(" ");
                int[] intSeq = Array.ConvertAll(strSeq, i => int.Parse(i));

                int profit = getMaxSubArray(intSeq, P);
                Console.WriteLine(profit > 0 ? profit : 0);
            }
            catch(Exception e)
            {
                Environment.Exit(0);
            }
        }

        private static int getMaxSubArray(int[] arr, int P)
        {
            int maxSum = arr[0] - P;
            int currentSum = maxSum;
            for (int i = 0; i < arr.Length; i++)
            {
                currentSum = Math.Max(currentSum + arr[i] - P, arr[i] - P);
                maxSum = Math.Max(currentSum, maxSum);
            }
            return maxSum;
        }
    }
}

```

Explanations provided by the candidate

Explanation of the problem: No explanation given

Explanation of the Solution: This problem contains a subproblem of a maximum sum of a subarray in a given array. The famous Kadene's algorithm can be used to resolve the sum. An offset of P can be subtracted to each element of the given input array. We can then calculate maxSubArray to fetch the maximum profit.

Proficiency Profile™

IKM TeckChek

MEDIUM

Chess

SHOWN AS	MEMORY LIMIT	CORRECTNESS	PLAGIARISM
Problem C	**1024 MB**	✓ **Accepted**	✓ **Not detected**
CPU TIME LIMIT	LANGUAGE USED	ATTEMPTS	EXPLANATIONS
1 second	**C#**	**3**	✓ **Added**

In chess the bishop is the chessman, which can only move diagonal. It is well known that bishops can reach only fields of one color but all of them in some number of moves (assuming no other figures are on the field). You are given two coordinates on a chess-field and should determine, if a bishop can reach the one field from the other and how. Coordinates in chess are given by a letter (’A’ to ’H’) and a number (1 to 8). The letter specifies the column, the number the row on the chessboard.

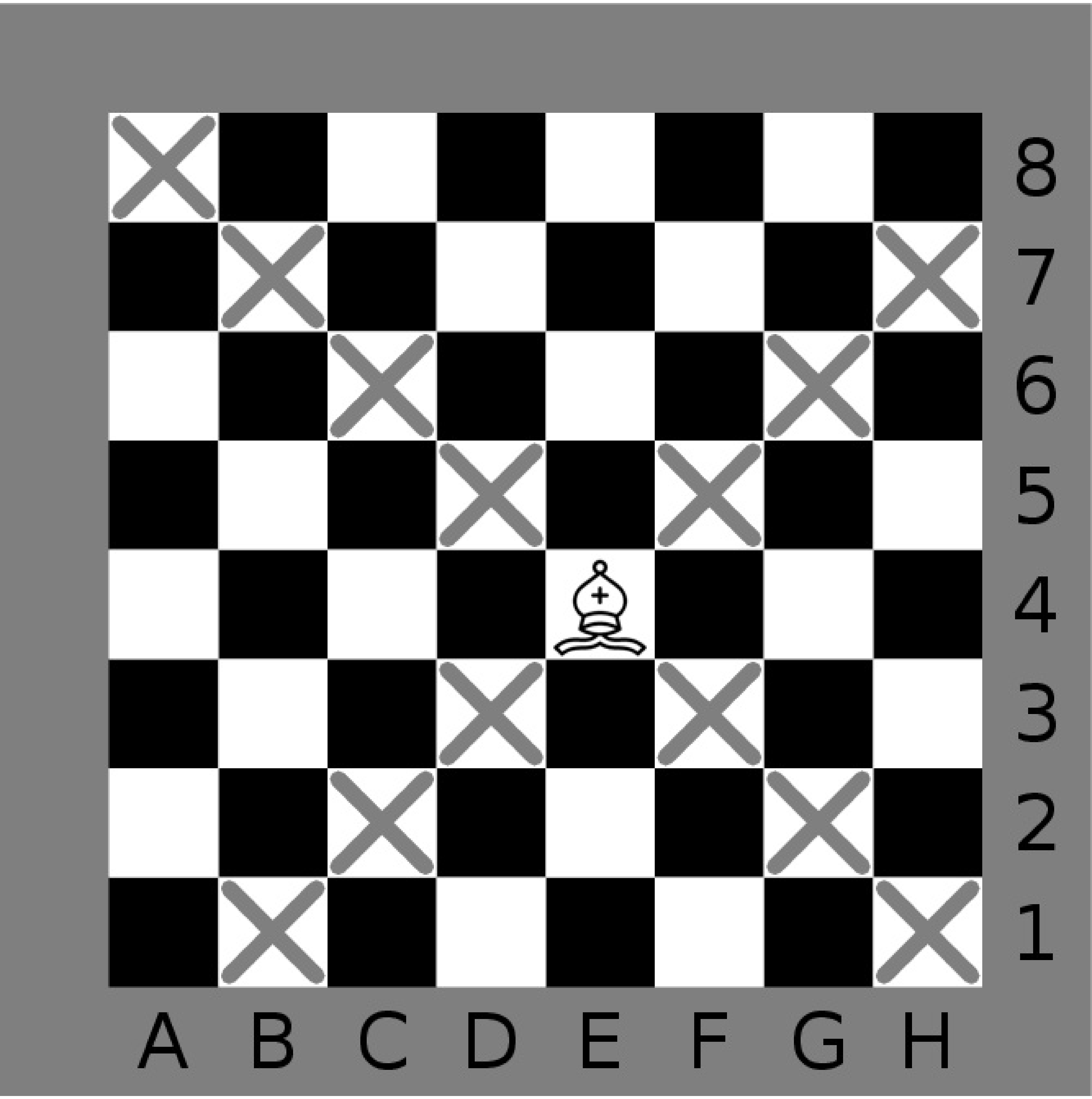


Figure 1: Chessboard, bishop and fields the bishop can reach in one move

Input

The input starts with the number of test cases. Each test case consists of one line, containing the start position X and end position Y. Each position is given by two space separated characters. A letter for the column and a number for the row. There are no duplicate test cases in one input.

Output

Output one line for every test case. If it's not possible to move a bishop from X to Y in any number of moves output 'Impossible'. Otherwise output one possible move sequence from X to Y. Output the number n of moves first (allowed to be 4 at most). Followed by $n + 1$ positions, which describe the path the bishop has to go. Every character is separated by one space. There are many possible solutions. Any with at most 4 moves will be accepted. Remember that in a chess move one chessman (the bishop in this case) has to change his position to be a valid move (i.e. two consecutive positions in the output must differ).

Sample Input 1

```
3
E 2 E 3
F 1 E 8
A 3 A 3
```

Sample Output 1

```
Impossible
2 F 1 B 5 E 8
0 A 3
```

5948198e11d24d20.cs

```
using System;
using System.Collections.Generic;

namespace BishopPath
{
    /*@author - Naveen*/
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                int noOfTestCases = Convert.ToInt32(Console.ReadLine());
                List<char[]> inputArr = new List<char[]>();

                for (int i = 1; i <= noOfTestCases; i++)                              //read inputs from the user
                {
                    char[] charArrayInput = Array.ConvertAll(Console.ReadLine().Split(" "), x => x[0]);
//convert the input string to character array
                    inputArr.Add(charArrayInput);
                }

                foreach (var input in inputArr)
                {
                    int startX, startY, endX, endY; char startRowLetter, endRowLetter;
//use the chess board as a two dimensional graph storing startX, startY, endX endY
                    startRowLetter = input[0];
                    startX = startRowLetter - 65;
                    startY = Convert.ToInt32(input[1].ToString()) - 1;

                    endRowLetter = input[2];
                    endX = endRowLetter - 65;
                    endY = Convert.ToInt32(input[3].ToString()) - 1;


                    if (startX == endX && startY == endY)                                                   //if the
starting and ending coordinates are same then return the same path
```

```
                    Console.WriteLine("{0} {1} {2}", "0", startRowLetter.ToString(), startY + 1);

                else if (Math.Abs(startX - endX) == Math.Abs(startY - endY))                //if the slope of the start and end coordinates is 1 or -1 (they are in the same line or diagnol)
                                                                                            // hence only one step is required in this scenario
                    Console.WriteLine("{0} {1} {2} {3} {4}", "1", startRowLetter.ToString(), startY + 1, endRowLetter.ToString(), endY + 1);

                else if ((startX + startY) % 2 != (endX + endY) % 2)                        //if starting coordinates and ending coordinates are in different color (white - black)
                                                                                            //no path exists between them
                    Console.WriteLine("Impossible");

                else
                {                                                                           //if the two points are not in same line, then it requires two steps to reach destination
                    int x = ((startY + startX) - (endY - endX)) / 2;
                    int y = x + (endY - endX);                                              //any source and destination can be reached in one/two paths if first is out of the chessboard calculate second one

                    if (x < 0 || x > 7 || y < 0 || y > 7)                                   //if previous mid-point is out of chessboard, calculate the second possible point
                    {
                        x = ((endY + endX) - (startY - startX)) / 2;
                        y = x + (startY - startX);
                    }
                    char x_temp = (char)(x + 65);
                    int y_temp = y + 1;

                    Console.WriteLine("{0} {1} {2} {3} {4} {5} {6}", "2", startRowLetter.ToString(), startY + 1, x_temp, y_temp, endRowLetter.ToString(), endY + 1);
                }
            }
            Environment.Exit(0);
        }
        catch (Exception e)
        {
            Environment.Exit(0);
        }
    }
}
```

```
}
```

Explanations provided by the candidate

Explanation of the problem: No explanation given

Explanation of the Solution: This problem can be solved using two ways - Graphs - with vertices and edges Math - with slope calculation I preferred the Mathematical linear equations and two-dimensions. Comments are provided along with the program. Please review.