I. Играйте в футбол!

|  |  |
| --- | --- |
| Ограничение времени | 2 секунды |
| Ограничение памяти | 256Mb |
| Ввод | стандартный ввод или input.txt |
| Вывод | стандартный вывод или output.txt |

Ася Вуткина — известный футбольный комментатор. Будучи профессионалом своего дела, Ася тщательно следит за всеми матчами всех европейских чемпионатов.

Благодаря накопленной информации, Ася может во время трансляции матча сообщить какую-нибудь интересную статистику, например: «Индзаги третий матч подряд забивает гол на 9-й минуте» или «Матерацци никогда не открывает счет в матче».

Но мозг Аси не безграничен, а помнить всю историю футбола просто невозможно. Поэтому Ася попросила вас написать программу, которая собирает статистику матчей и умеет отвечать на некоторые запросы, касающиеся истории футбола.

Информация о матче сообщается программе в следующей форме:

"<Название 1-й команды>" - "<Название 2-й команды>" <Счет 1-й команды>:<Счет 2-й команды>

<Автор 1-го забитого мяча 1-й команды> <Минута, на которой был забит мяч>'

<Автор 2-го забитого мяча 1-й команды> <Минута, на которой был забит мяч>'

...

<Автор последнего забитого мяча 1-й команды> <Минута, на которой был забит мяч>'

<Автор 1-го забитого мяча 2-й команды> <Минута, на которой был забит мяч>'

...

<Автор последнего забитого мяча 2-й команды> <Минута, на которой был забит мяч>'

Запросы к программе бывают следующих видов:  
  
Total goals for <Название команды>

— количество голов, забитое данной командой за все матчи.

Mean goals per game for <Название команды>

— среднее количество голов, забиваемое данной командой за один матч. Гарантирутся, что к моменту подачи такого запроса команда уже сыграла хотя бы один матч.

Total goals by <Имя игрока>

— количество голов, забитое данным игроком за все матчи.

Mean goals per game by <Имя игрока>

— среднее количество голов, забиваемое данным игроком за один матч его команды.

Гарантирутся, что к моменту подачи такого запроса игрок уже забил хотя бы один гол.

Goals on minute <Минута> by <Имя игрока>

— количество голов, забитых данным игроком ровно на указанной минуте матча.

Goals on first <T> minutes by <Имя игрока>

— количество голов, забитых данным игроком на минутах с первой по *T*-ю включительно.

Goals on last <T> minutes by <Имя игрока>

— количество голов, забитых данным игроком на минутах с *(91 - T)*-й по 90-ю включительно.

Score opens by <Название команды>

— сколько раз данная команда открывала счет в матче.

Score opens by <Имя игрока>

— сколько раз данный игрок открывал счет в матче.

Формат ввода

Входной файл содержит информацию о матчах и запросы в том порядке, в котором они поступают в программу Аси Вуткиной.

Во входном файле содержится информация не более чем о 100 матчах, в каждом из которых забито не более 10 голов. Всего в чемпионате участвует не более 20 команд, в каждой команде не более 10 игроков забивают голы.

Все названия команд и имена игроков состоят только из прописных и строчных латинских букв и пробелов, а их длина не превышает 30. Прописные и строчные буквы считаются различными. Имена и названия не начинаются и не оканчиваются пробелами и не содержат двух пробелов подряд. Каждое имя и название содержит хотя бы одну букву.

Минута, на которой забит гол — целое число от 1 до 90 (про голы, забитые в дополнительное время, принято говорить, что они забиты на 90-й минуте).

Для простоты будем считать, что голов в собственные ворота в европейских чемпионатах не забивают, и на одной минуте матча может быть забито не более одного гола (в том числе на 90-й). Во время чемпионата игроки не переходят из одного клуба в другой.

Количество запросов во входном файле не превышает 500.

Формат вывода

Для каждого запроса во входном файле выведите ответ на этот запрос в отдельной строке. Ответы на запросы, подразумевающие нецелочисленный ответ, должны быть верны с точностью до трех знаков после запятой.

Пример 1

| **Ввод**  Скопировать ввод | **Вывод**  Скопировать вывод |
| --- | --- |
| "Juventus" - "Milan" 3:1  Inzaghi 45'  Del Piero 67'  Del Piero 90'  Shevchenko 34'  Total goals for "Juventus"  Total goals by Pagliuca  Mean goals per game by Inzaghi  "Juventus" - "Lazio" 0:0  Mean goals per game by Inzaghi  Mean goals per game by Shevchenko  Score opens by Inzaghi | 3  0  1.0  0.5  1.0  0 |

Пример 2

| **Ввод**  Скопировать ввод | **Вывод**  Скопировать вывод |
| --- | --- |
| Total goals by Arshavin | 0 |

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

public struct Point

{

public int dX;

public int dY;

public Point(int x, int y)

{

dX = x;

dY = y;

}

public void Print()

{

Console.WriteLine("\t\tPoint coordinates: X = {0}, Y = {1}", dX, dY);

}

public override int GetHashCode()

{

return dX.GetHashCode() \* dY.GetHashCode();

}

}

public struct PointWhisLen

{

public int dX;

public int dY;

public int X;

public int Y;

public PointWhisLen(int dx, int dy, int x, int y)

{

dX = dx;

dY = dy;

X = x;

Y = y;

}

public void Print1()

{

Console.WriteLine("Point dif: X = {0}, Y = {1}", dX, dY);

}

public override int GetHashCode()

{

return X.GetHashCode() \* Y.GetHashCode() \* dY.GetHashCode();

}

}

class Program

{

static void Main()

{

StreamReader reader = new StreamReader("input.txt");

int N = Convert.ToInt32(reader.ReadLine());

Dictionary<Point, HashSet<PointWhisLen>> start = new Dictionary<Point, HashSet<PointWhisLen>>();

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

{

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

int x1 = int.Parse(input[0]);

int y1 = int.Parse(input[1]);

int x2 = int.Parse(input[2]);

int y2 = int.Parse(input[3]);

int dX = Math.Abs(x2 - x1);

if ((x1 <= x2 && y1 >= y2) || (x1>=x2 && y2>=y1)) dX = -dX;

int dY = Math.Abs(y2 - y1);

int lenX = x1 + x2, lenY = y1 + y2;

Point point = new Point(dX, dY);

PointWhisLen pointWhisLen = new PointWhisLen(dX, dY, lenX, lenY);

if (start.ContainsKey(point)) start[point].Add(pointWhisLen);

else start.Add(point, new HashSet<PointWhisLen>(){pointWhisLen});

}

Dictionary<Point, HashSet<PointWhisLen>> end = new Dictionary<Point, HashSet<PointWhisLen>>();

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

{

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

int x1 = int.Parse(input[0]);

int y1 = int.Parse(input[1]);

int x2 = int.Parse(input[2]);

int y2 = int.Parse(input[3]);

int dX = Math.Abs(x2 - x1);

if ((x1 <= x2 && y1 >= y2) || (x1>=x2 && y2>=y1)) dX = -dX;

int dY = Math.Abs(y2 - y1);

int lenX = x1 + x2, lenY = y1 + y2;

Point this\_point = new Point(dX, dY);

if (start.ContainsKey(this\_point))

{

foreach (var spichka in start[this\_point])

{

Point this\_len = new Point(spichka.X - lenX, spichka.Y - lenY);

if (end.ContainsKey(this\_len))

{

end[this\_len].Add(spichka);

}

else

{

end.Add(this\_len, new HashSet<PointWhisLen>(){spichka});

}

}

}

}

int Max = 0;

foreach (var VARIABLE in end)

{

if (end[VARIABLE.Key].Count > Max) Max = end[VARIABLE.Key].Count;

}

Console.WriteLine(N - Max);

}

}