Задача A. Есть контакт!

Имя входного файла: `стандартный ввод`
Имя выходного файла: `стандартный вывод`
Ограничение по времени: 2 секунды на 10 ходов
Ограничение по памяти: 64 мегабайта

В этой игре не будет никаких легенд, войн, да и вообще хоть сколько-нибудь значительного вступления. Перейдём сразу к делу. Игра происходит на доске 20 × 20 клеток. Два игрока по очереди захватывают себе клетки. Цель первого игрока — соединить своими клетками верхнюю и нижнюю границы доски, цель второго — соединить левую и правую границу. Две границы поля называются соединёнными, если существует путь от одной границы до другой по клеткам соответствующего игрока. При этом переходить можно только по клеткам, соседним по стороне.

Формат входного файла

Перед каждым ходом в стандартный поток ввода Вашей программе будет передано текущее состояние поля, 20 строк по 20 чисел в каждой. Завершаться передача состояния будет пустой строкой. Каждое из этих чисел a_{ij} соответствует одной клетке поля. Если $a_{ij} = 1$, то эта клетка захвачена Вами. Если $a_{ij} = 2$, эта клетка захвачена Вашим противником. Если же $a_{ij} = 0$, клетка еще никем не захвачена. Перед Вашим ходом поле будет повёрнуто таким образом, что Вам будет требоваться соединить верхнюю и нижнюю границы (даже если Вы являетесь вторым игроком).

Формат выходного файла

На каждом ходу Вам требуется вывести номер строки и столбца клетки, которую Вы хотите захватить. Строки и столбцы нумеруются с нуля, строки — сверху вниз, столбцы — слева направо. После вывода хода выведите пустую строку. Не забывайте сбрасывать буфер выходного потока после каждого хода. В C/C++ это можно сделать с помощью `fflush(stdout);`, в Pascal с помощью `flush(output);`

Примеры

`стандартный ввод`

```
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
2 2 2 2 2 2 2 2 2 0 2 2 2 2 2 2 2 2 2 2
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
```

`стандартный вывод`

```
11 9
```

В примере игрок, делающий ход, выигрывает.

Пример реализованной стратегии, на каждом ходу делающей ход в случайную клетку (C++):

```
#include <iostream>
#include <fstream>
#include <ctime>
#include <cstdlib>

using namespace std;

const int SZ = 20;
int field[SZ][SZ];
int numfree;
int freerow[SZ * SZ];
int freecol[SZ * SZ];

int main()
{
    srand((unsigned)time(NULL));
    int i, j;
    while (1)
    {
        for (i = 0; i < SZ; i++)
        {
            for (j = 0; j < SZ; j++)
            {
                scanf("%d", &field[i][j]);
            }
        }
        numfree = 0;
        for (i = 0; i < SZ; i++)
        {
         for (j = 0; j < SZ; j++)
         {
         if (field[i][j] == 0)
         {
         freerow[numfree] = i;
         freecol[numfree] = j;
         numfree++;
         }
         }
        }
        if (numfree == 0) break;
        int pos = rand() % numfree;
        printf("%d %d\n\n", freerow[pos], freecol[pos]);
        fflush(stdout);
    }
    // чтобы программа не завершилась раньше времени
    scanf("%d", &i);
    return 0;
}
```

Пример реализованной стратегии, на каждом ходу делающей ход в случайную клетку (Pascal):

```
const SZ = 20;
var field : array[1..SZ, 1..SZ] of integer;
i, j : integer;
freerow, freecol : array[1..SZ * SZ] of integer;
numfree, pos : integer;
begin
randomize;
while true do begin
        for i := 1 to SZ do begin
         for j := 1 to SZ do begin
         read(field[i, j]);
         end;
        end;
        numfree := 0;
        for i := 1 to SZ do begin
         for j := 1 to SZ do begin
         if (field[i, j] = 0) then begin
         inc(numfree);
         freerow[numfree] := i;
         freecol[numfree] := j;
         end;
         end;
        end;
        if (numfree = 0) then break;
        pos := trunc(random() * 1000) mod numfree + 1;
        writeln(freerow[pos] - 1, ' ', freecol[pos] - 1);
        writeln;
        flush(output);
    end;
    read(i);
end.
```

Задача B. Перестрелка

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды на 10 ходов
Ограничение по памяти:	64 мегабайта

Много лет идёт война между Красными и Синими. Сначала они воевали при помощи палок и камней, потом в ход пошли мечи и щиты, потом ружья. И вот теперь основной боевой единицей в этой бессмысленной и беспощадной войне является танк. Каждый танк экипирован энергетической пушкой, стреляющей сгустками энергии и батареей, которая может хранить до 100 единиц энергии. В течение боя экипаж танка может выбирать, какое количество энергии потратить на выстрел. К сожалению, конструкция танка такова, что через 10 выстрелов пушка перегревается и бой на этом заканчивается.

Вам предлагается написать программу управления танком, выбирающую, какое количество энергии требуется потратить на каждый выстрел. При перестрелке двух танков, они одновременно делают 10 выстрелов. Выстрелы двух танков сталкиваются в воздухе. При столкновении выстрел большей мощности нейтрализует менее мощный, после чего продолжает своё движение. Когда выстрел попадает в танк, он наносит танку одну единицу урона (вне зависимости от мощности выстрела). Если сталкиваются два выстрела одинаковой мощности, они взаимоуничтожаются. Вашей задачей будет нанести вражескому танку как можно большее повреждение за 10 выстрелов. Помните, что у вас всего 100 единиц энергии и расходуйте её экономно.

Формат входного файла

Перед каждым ходом в стандартный поток ввода Вашей программе будет передано состояние предыдущего выстрела, описываемое двумя числами: количеством энергии, которое потратил Ваш танк на выстрел и количеством энергии, которое потратил вражеский танк на выстрел. Завершаться передача состояния будет пустой строкой. Перед первым ходом Вам будут переданы два нуля.

Формат выходного файла

На каждом ходу Вам требуется вывести количество энергии, которое Вы хотите потратить на очередной выстрел. После вывода хода выведите пустую строку. Не забывайте сбрасывать буфер выходного потока после каждого хода. В C/C++ это можно сделать с помощью `fflush(stdout);`, в Pascal с помощью `flush(output);`

Примеры

стандартный ввод
10 10
стандартный вывод
90

В примере танк расходует всю свою энергию за два хода: 10 единиц на предыдущем ходу и 90 на текущем. Будьте внимательны: Вам не следует завершать свою программу после 10 ходов, система завершит её самостоятельно. Используйте функции чтения для приостановки программы после 10-го хода.

Пример реализованной стратегии, на каждом ходу расходующей случайное количество энергии (C++):

```
#include <iostream>
#include <fstream>
#include <ctime>
#include <cstdlib>

using namespace std;


int main()
{
    srand((unsigned)time(NULL));
    int energy = 100;
    int i;
    int we, them;
    for (i = 0; i < 10; i++)
    {
     scanf("%d%d", &we, &them);
     int cur = rand() % (energy + 1);
     printf("%d\n\n", cur);
     energy -= cur;
     fflush(stdout);
    }
    scanf("%d", &we);
    return 0;
}
```

Пример реализованной стратегии, на каждом ходу расходующей случайное количество энергии (Pascal):

```
var energy : integer;
i, cur, we, them : integer;
begin
randomize;
energy := 100;
for i := 1 to 10 do begin
readln(we, them);
cur := trunc(random() * 1000) mod (energy + 1);
writeln(cur);
writeln;
        flush(output);
        energy := energy - cur;
    end;
    readln(we);
end.
```

Задача C. Змейка

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды на 10 ходов
Ограничение по памяти:	64 мегабайта

Многие наверное помнят игру “Змейка”, широко распространённую в китайских игровых консолях наряду с “Тетрисом” и в телефонах одной известной фирмы. В этой задаче Вам предлагается написать программу, играющую в змейку.

Игровое поле представляет собой матрицу клеток размером 30×30. В каждой клетке поля может находиться одна из двух змеек или яблоко. Каждая змейка состоит некоторого количества клеток, пронумерованных, начиная с единицы. На каждом ходу змейка выбирает направление движения, после чего первая клетка змейки сдвигается в этом направлении, а каждая последующая клетка занимает место предыдущей. Если первая клетка змейки оказывается на клетке с яблоком, змейка становится на одну клетку длиннее. Новая клетка добавляется к змейке в конец после сдвига (то есть на ту клетку, где была последняя клетка змейки перед ходом).

Яблоки появляются на поле случайным образом, с частотой примерно 1 яблоко на 10 ходов (здесь ходом считается ход одной змейки и ответ другой).

Изначально каждая змейка состоит из одной клетки, расположенной в клетке $(4, 4)$ или $(25, 25)$ (нумерация клеток с нуля).

Если в результате хода змейки ее голова должна быть сдвинута за границу поля или на клетку, занятую какой-либо змейкой, ходящая змейка объявляется проигравшей. Если в результате хода змейки её длина достигает 250 клеток, змейка объявляется победившей.

Формат входного файла

Перед каждым ходом в стандартный поток ввода Вашей программе будет передано текущее состояние поля, 30 строк по 30 чисел в каждой. Завершаться передача состояния будет пустой строкой. Каждое из этих чисел a_{ij} соответствует одной клетке поля. Если $0 < a_{ij} \leq 250$, то в этой клетке поля находится клетка Вашей змейки с номером a_{ij}. Если $a_{ij} < 0$, то в этой клетке находится клетка змейки соперника с номером $-a_{ij}$. Если $a_{ij} = 1000$, в клетке находится яблоко. Если же $a_{ij} = 0$, клетка пуста.

Формат выходного файла

На каждом ходу Вам требуется вывести направление, в котором Вы хотите переместить свою змейку. Каждое из четырёх направлений обозначается одним символом: `U` — вверх, `L` — влево, `R` — вправо, `D` — вниз. После вывода хода выведите пустую строку. Не забывайте сбрасывать буфер выходного потока после каждого хода. В C/C++ это можно сделать с помощью `fflush(stdout);`, в Pascal с помощью `flush(output);`

Примеры

стандартный ввод

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 3 2 1 1000 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -2 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -3 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -6 -5 -4 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -7 -8 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

стандартный вывод

```
R
```

В примере змейка, совершающая ход, удлинится на одну единицу после хода.

Пример реализованной стратегии, на каждом ходу делающей ход в случайном направлении (C++):

```
#include <iostream>
#include <fstream>
#include <ctime>
#include <cstdlib>

using namespace std;

const int SZ = 30;
int field[SZ][SZ];
const char dir[5] = "LURD";

int main()
{
    srand((unsigned)time(NULL));
    int i, j;
    while (1)
    {
        for (i = 0; i < SZ; i++)
        {
            for (j = 0; j < SZ; j++)
            {
                scanf("%d", &field[i][j]);
            }
        }
        printf("%c\n\n", dir[rand() % 4]);
        fflush(stdout);
    }
    return 0;
}
```

Пример реализованной стратегии, на каждом ходу делающей ход в случайную клетку (Pascal):

```
const SZ = 30;
const dir : array[0..3] of char = ('L', 'R', 'U', 'D');
var field : array[1..SZ, 1..SZ] of integer;
i, j : integer;
begin
randomize;
while true do begin
        for i := 1 to SZ do begin
         for j := 1 to SZ do begin
         read(field[i, j]);
         end;
        end;
        writeln(dir[trunc(random() * 1000) mod 4]);
        writeln;
        flush(output);
    end;
end.
```