ФБЦ. 7-9 класс

В задаче рассмотрим факториальную буквенно-цифровую систему счисления (далее будем писать сокращённо – ФБЦ-систему). В ней используются цифры d_i, $i = 0 \dots 35$: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z. Строчные буквы не являются цифрами, используются только заглавные (прописные). Каждой цифре d_i приписано её числовое значение $val(d_i) = i$, так $val(0) = 0$, ..., $val(9) = 9$, $val(A) = 10$, ..., $val(Z) = 35$. Запись в ФБЦ-системе $d_{i_n} d_{i_{n-1}} d_{i_{n-2}} \dots d_{i_2} d_{i_1}$, где $0 \le val(d_{i_k}) \le k$ и $1 \le k \le n \le 35$, означает беззнаковое число, равное $val(d_{i_n}) * n! + val(d_{i_{n-1}}) * (n-1)! + \dots + val(d_{i_2}) * 2! + val(d_{i_1}) * 1!$. Запись числа в ФБЦ-системе содержит не более чем 35 цифр и не содержит знака числа, то есть не все числа могут быть в ней записаны. На k-ой позиции в ФБЦ-записи (нумерация позиций ведётся справа налево, начиная с 1) допускается указывать цифру, числовое значение которой $val(d_{i_k})$ не превышает k и не меньше нуля. На самой правой позиции может быть либо 0, либо 1. На второй справа позиции может быть либо 0, либо 1, либо 2. И так далее. Допускается наличие незначащих нулей в левых позициях ФБЦ-записи. Незначащим является любой нуль, стоящий до первой стоящей слева ненулевой цифры, или, если записано нулевое число, то все нули, кроме самого правого. Например, десятичное число $100_{\mathbf{10}} = 96 + 0 + 4 + 0 = 4 * 4! + 0 * 3! + 2 * 2! + 0 * 1! = 4020_{\mathbf{ФБЦ}}$. Здесь приписанное снизу $_{\mathbf{ФБЦ}}$ помечает запись числа в ФБЦ-системе. То же самое число может быть записано с незначащими нулями. Например, $0004020_{\mathbf{ФБЦ}}$. Здесь три незначащих нуля.

Составьте программу, которая принимает на вход в первой строке десятичное число K – положительное натуральное число ($1 \le K \le 35$); во второй строке десятичное число N – положительное натуральное число ($1 \le N \le 50000$) – длину последовательности ФБЦ-чисел, и в последующих N строках – записи чисел X_i в ФБЦ-системе счисления, где $1 \le i \le N$. Программа находит элемент последовательности ФБЦ-чисел, который кратен $K!$ и при этом отстоит как можно дальше от начала последовательности. Программа выводит в первой строке запись найденного элемента последовательности в ФБЦ-системе без незначащих нулей, а во второй – номер этого элемента, записанный десятичным натуральным числом без знака. Нумерация элементов последовательности ведётся по порядку их следования от начала к концу, начальный элемент имеет номер 1. Если искомый элемент отсутствует, то программа выводит единственную строку с числом -1.

Формат ввода: В первой строке содержится десятичное число K ($1 \le K \le 35$). Во второй строке содержится десятичное число N — длина последовательности ($1 \le N \le 50000$). В следующих N строках содержатся записи чисел X_i в ФБЦ-системе счисления, где $1 \le i \le N$. В записи числа X_i используются только десятичные цифры $(0, \dots, 9)$ и заглавные латинские буквы $(A, \dots, Z)$.

Формат вывода: Если среди введённых чисел X_i отсутствуют те, которые кратны $K!$, то программа в первой строке выводит -1 и ничего более не выводит. Иначе в первой строке выводится запись числа X_i в ФБЦ-системе без незначащих нулей, и это X_i таково, что оно кратно $K!$ и находится дальше от начала последовательности, чем другие элементы последовательности, кратные $K!$. Во второй строке выводится беззнаковое десятичное натуральное число, равное i – номеру элемента последовательности X_i, выведенного в первой строке. Нумерация элементов последовательности ведётся по порядку их следования от начала к концу, начальный элемент имеет номер 1.

Пример №1:
ввод:
2
1
4000
вывод:
4000
1
Пример №2:
ввод:
35
3
00000000000000000000000000000000000
0000000000000000

0000000
вывод:
0
3
Пример №3:
ввод:
4
4
$NKKJJIIHHGGFFEEDDCCBBAA554433222000$
$LKKJJIIHHGGFFEEDDCCBBAA554433220000$
$NKKJJIIHHGGFFEEDDCCBBAA554433222000$
$MKKJJIIHHGGFFEEDDCCBBAA554433221100$
вывод:
$NKKJJIIHHGGFFEEDDCCBBAA554433222000$
3

Решение

В решении можно запрограммировать следующие подзадачи: 1) считывание очередного числа и представление его в виде строки из 35 символов с незначащими нулями, дополняющими считанную запись слева до 35 цифр; 2) определение кратности $K!$ считанного числа на основе подсчёта количества нулей, которыми заканчивается справа запись числа, и того факта, что ноль кратен любому ненулевому числу; 3) определение последнего числа, кратного $K!$, среди считанных чисел. Для решения третьей подзадачи достаточно одного прохода по последовательности, в котором совмещены построчный ввод чисел и их обработка. Следует хранить текущего кандидата на ответ (последнее число, кратное $K!$, среди всех чисел, которые программа успела считать) и его номер. Очередное число после считывания проверяется на кратность. При удачной проверке кандидат на ответ обновляется. Иначе делается переход к обработке следующего числа. По окончании обработки выводится искомое число и его номер (или -1 при неудачном поиске).

Код возможного решения

```
program FBC0709 (input, output);
const   MAXN = 35;
type    fbcnumber = array [1..MAXN] of char;
        answer = record no : word; number : fbcnumber end;
var     CURNUM : fbcnumber;
        K, N, I : word;
        CURANSWER : answer;
procedure readnumber(var FBCNUM : fbcnumber);
var     S : string;
        I, J : byte;
begin   readln(S);
        J := MAXN;
        for I := Length(S) downto 1 do begin
            FBCNUM[J] := S[I];
            J := J - 1;
        end;
        for I := J downto 1 do FBCNUM[I] := '0';
end;
function isdividable(var FBCNUM : fbcnumber; K : word) : boolean;
var     I : word;
begin   I := MAXN;
        while (I > 0) and (FBCNUM[I] = '0') do I := I - 1;
        isdividable := (I = 0) or (I <= (MAXN + 1 - K))
end;
begin   readln(K);
        readln(N);
        with CURANSWER do begin
```

```
            no := 65535;
            for I := 1 to MAXN do number[I] := '0';
            for I := 1 to N do begin
                readnumber(CURNUM);
                if isdividable(CURNUM, K) then begin no := I; number := CURNUM end;
            end;
            if (no = 65535) then writeln(-1)
            else begin
                I := 1;
                while (number[I] = '0') and (I < MAXN) do I := I + 1;
                while I <= MAXN do begin write(number[I]); I := I + 1 end;
                writeln;
                writeln(no);
            end;
        end;
end.
```

Большое ФБЦ-число. 7-9 класс

В задаче рассмотрим факториальную буквенно-цифровую систему счисления (сокращённо ФБЦ-систему) из задачи "ФБЦ".

Составьте программу, которая принимает на вход в первой строке десятичное число N – положительное натуральное число ($1 \leq N \leq 50000$) – длину последовательности символов, и во второй строке – саму эту последовательность, составленную из N символов из таблицы ASCII. В последовательности не используются управляющие символы, такие как перевод строки и другие. Программа находит максимальное ФБЦ-число, запись которого можно составить из символов последовательности. При составлении записи числа можно использовать символы из введённой последовательности в количестве, которое не больше количества их вхождений во введённую последовательность. Например, если цифра 0 входит в последовательность дважды, то в составленной записи искомого числа 0 встречается не более чем два раза. Нельзя менять регистр символов (т. е. делать строчные буквы заглавными или наоборот).

Программа выводит в единственной строке запись найденного числа в ФБЦ-системе без незначащих нулей. Если введённые символы таковы, что из них не получается составить запись какого-либо числа в ФБЦ-системе, то программа выводит единственную строку с числом -1.

Формат ввода: В первой строке содержится десятичное число N ($1 \leq N \leq 50000$). Во второй строке содержится последовательность, составленная из N символов из таблицы ASCII, но без управляющих символов, таких как перевод строки и другие.

Формат вывода: Если среди введённой последовательности недостаточно символов, подходящих для записи ФБЦ-числа, то программа в первой строке выводит -1 и ничего более не выводит. Иначе в единственной строке выводится максимальное ФБЦ-число, запись которого можно составить из символов последовательности. При составлении записи числа используются символы из введённой последовательности в количестве, которое не больше количества их вхождений во введённую последовательность. В записях ФБЦ-чисел используются только десятичные цифры ($0, \ldots, 9$) и заглавные латинские буквы ($A, \ldots, Z$). Выводится запись найденного числа в ФБЦ-системе без незначащих нулей.

Пример №1:

ввод:

65

01020304056789ABCDEFGHIJKLMNOPQRSTUVWXYabcdefghijklmnopqrstuvwxyz

вывод:

YXWVUTSRQPONMLKJIHGFEDCBA9876543210

Пример №2:

ввод:

2

22

вывод:

-1

Пример №3:

ввод:

15

*aA334422551100

вывод:

A554433221100

Решение

В решении можно запрограммировать следующие подзадачи: 1) определение количества вхождений каждого из символов – цифр и букв, используемых в записи ФБЦ-чисел, – $0, \ldots, 9, A, \ldots, Z$ в данную последовательность; 2) определение максимально возможной длины искомого ФБЦ-числа; 3) определение искомого числа при известной его длине. Для решения первой подзадачи достаточно одного прохода по последовательности. Для решения второй подзадачи можно использовать жадный алгоритм. С учётом правил записи ФБЦ-чисел следует суммировать количество вхождений сначала цифры 0, потом 1 и т. д., останавливаясь, если числовое значение цифры превышает текущую сумму более чем на 1. Для решения третьей подзадачи также можно использовать жадный алгоритм. Зная длину искомого числа, начиная со старшего его разряда, следует помещать в разряд цифру, которая является наибольшей среди допустимых правилами

записи ФБЦ-чисел. После использования цифры, количество её вхождений уменьшается на 1. Далее осуществляется переход к соседнему более младшему разряду. По окончании обработки выводится составленное число (или -1 при неудачном поиске).

Код возможного решения

```
program MAXFBC0709 (input, output);
const   MAXN = 35;
type    fbcnumber = array [1..MAXN] of char;
var     ANSWER : fbcnumber;
        N : word;
        DIGITS : array [0..MAXN] of word;
        C : char;
        I, J : byte;
function num2char(I : byte) : char;
begin   if (I >= 0) and (I <= 9) then num2char := chr(I + ord('0'))
        else if (I >= 10) and (I <= MAXN) then num2char := chr(I - 10 + ord('A'))
        else num2char := '/'
end;
function char2num(C : char) : byte;
begin   if (C >= '0') and (C <= '9') then char2num := ord(C) - ord('0')
        else if (C >= 'A') and (C <= 'Z') then char2num := ord(C) - ord('A') + 10
        else char2num := 255
end;
procedure readsequence(N : word);
var     C : char;
        I : word;
        J : byte;
begin   for I := 1 to N do begin
            read(C);
            J := char2num(C);
            if (J <> 255) then DIGITS[J] := DIGITS[J] + 1
        end;
end;
function calclenght() : word;
var     L, I : word;
        C : char;
begin   L := 0;
        C := '0';
        I := char2num(C);
        while (I <> 255) and ((I - 1) <= L) and ( L < MAXN) and (C <= '9') do
            begin L := L + DIGITS[I]; C := succ(C); I := char2num(C) end;
        C := 'A';
        I := char2num(C);
        while (I <> 255) and ((I - 1) <= L) and ( L < MAXN) and (C <= 'Z') do
            begin L := L + DIGITS[I]; C := succ(C); I := char2num(C) end;
        if L > MAXN then L := MAXN;
        calclenght := L;
end;
begin   readln(N);
        for I := 0 to MAXN do DIGITS[I] := 0;
        readsequence(N);
        J := calclenght();
        if J = 0 then writeln(-1)
        else begin
            for I := 1 to MAXN do ANSWER[I] := '0';
            I := MAXN;
```

```
            while J > 0 do begin
                if I > J then I := J;
                while (I >= 0) and (I <= MAXN) and (DIGITS[I] = 0)
                do I := I - 1;
                if (I >= 0) and (I <= MAXN) and (DIGITS[I] > 0) then begin
                    ANSWER[J] := num2char(I);
                    DIGITS[I] := DIGITS[I] - 1;
                end;
                J := J - 1
            end;
            I := MAXN;
            while (ANSWER[I] = '0') and (I > 1) do I := I - 1;
            while I >= 1 do begin write(ANSWER[I]); I := I - 1 end;
            writeln
        end;
end.
```

Шнурок

На столе лежит деревянный прямоугольник с двумя линиями отверстий. Отверстий в каждой линии равное количество, для каждого отверстия есть его "пара" напротив. Эти отверстия зашнуровали одним шнурком, при этом:

- Один стежок (кусок шнурка между двумя отверстиями) может идти только в отверсиями напротив или по диагонали в 45 градусов (на 1 деление выше или ниже).
- Стежки не могут совпадать (если шнурок прошел один раз между двумя выбранными отверстиями, он не может пройти там второй раз)

После этого сделали две фотографии -- лицевой и изнаночной стороны шнуровки. Могут ли две выданные вам фотографии быть вариантами одной и той же шнуровки?

Формат входных данных:

В первой строке написан X ($1 \le X \le 100$) - количество отверстий в одной линии. Следующей строкой идет строка FACE. В последующих строках идет описание лицевой шнуровки - из какого отверстия в какое идет шнурок (нумерация отверстий начинается с 1). После идет строка BACK и идет описание изнаночной стороны шнуровки. После окончания описания идет строка END.

Формат выходных данных:

Выведите YES, если такая шнуровка может существовать и NO в ином случае.

входные данные:

```
2
FACE
1 1
2 2
BACK
1 2
2 1
END
```

выходные данные:

```
YES
```

входные данные:

```
3
FACE
1 1
2 2
3 3
BACK
1 2
2 1
END
```

выходные данные:

NO

Сеть

В одном большом главном здании есть N-комнат, которые связаны M-проводами (каналами связи) друг с другом. По этим каналам передается информация. Между каждой парой комнат информация может передаться, в том числе с использование промежуточных комнат. Некто решил разрушить связность сети, но его ресурсы на разрушение связности крайне ограничены. Вам требуется найти минимальное число проводов между комнатами, которые нужно разрезать, чтобы связность сети разрушилась.

Формат входных данных:

В первой строке задаётся число комнат N (1 < N <100) и число проводов M. В следующих M строках описание проводов - пара u_i, v_i комнат, которые соединены i-м проводом. Комнаты нумеруются от 1 до N. Между парой комнат может быть несколько проводов, комната может быть связана сама с собой.

Формат выходных данных

Выведите в первой строке минимальное число проводов, которые надо разрушить. Затем в соответствующем числе строк описание проводов в виде пары комнат которые им соединялись. В каждой паре номер должны быть упорядочены по возрастанию. Список пар должен быть упорядочен по возрастанию лексографически. В случае различных ответов для минимального числа проводов выведите любой из них.

входные данные:

```
2 1
1 2
```

выходные данные:

```
1
1 2
```

входные данные:

```
3 4
1 2
1 3
2 3
1 2
```

выходные данные:

```
2
1 3
2 3
```

Замощение

Плоскость замощена треугольными и квадратными плитками как показано на рисунке.

Позиция каждой плитки описывается парой координат a,b как показано на рисунке.

Таракан может переползти с одной плитки на другую, если эти две плитки имеют общее ребро. Если плитки имеют общую вершину, таракан переползти с одной плитки на другую не может.

По координатам начальной и конечной плиток найдите минимальное количество плиток, которые должен посетить таракан, чтобы доползти от начальной до конечной плитки. Начальная и конечная плитки учитываются в их количестве.

Формат входных данных

Во входных данных сначала задаются координаты a1,b1 начальной плитки, затем координаты a2,b2 конечной плитки. Все координаты целые числа и находятся в отрезке [-2000000000, 2000000000].

Формат выходных данных

Выведите минимальное количество плиток, которые нужно посетить, чтобы добраться от начальной плитки до конечной.

входные данные:

```
1 -3 1 -3
```

выходные данные:

```
1
```

входные данные:

```
0 2 1 1
```

выходные данные:

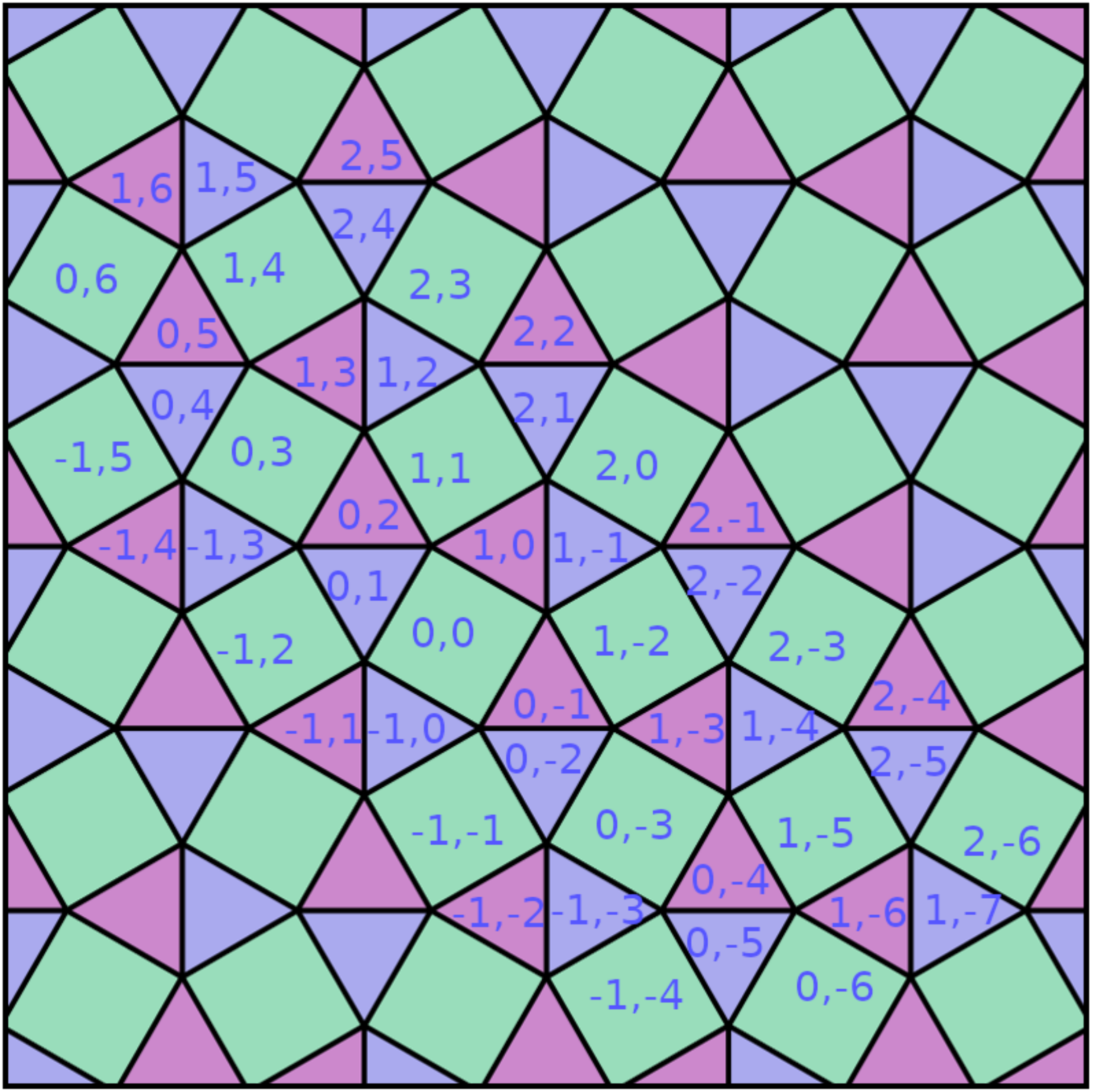
```
2
```

входные данные:

```
0 1 2 1
```

выходные данные:

```
4
```


2,5
1,6
1,5
2,4
0,6
1,4
2,3
0,5
2,2
1,3
1,2
0,4
2,1
-1,5
0,3
1,1
2,0
0,2
2.-1
-1,4
-1,3
1,0
1,-1
0,1
2,-2
-1,2
0,0
1,-2
2,-3
0,-1
2,-4
-1,1
-1,0
1,-3
1,-4
0,-2
2,-5
-1,-1
0,-3
1,-5
2,-6
0,-4
-1,-2
-1,-3
1,-6
1,-7
0,-5
-1,-4
0,-6

Решения задач участников оценивались следующим образом:

- 1 задача — до 100 баллов
- 2 задача — до 100 баллов
- 3 задача — до 100 баллов
- 4 задача — до 100 баллов
- 5 задача — до 100 баллов

Итоговая сумма баллов, набранная участником, переводилась в стобальную систему

Настройки тестового сервера и наборы тестов для проверки решений участников опубликованы по адресу: https://ejudge.cs.msu.ru/lom2022.tgz