Sapientia
Hungarian University of Transylvania

Sapientia ECN International Programming Contest – 2013

PROBLEMS

Târgu Mureş / Marosvásárhely, 11th May 2013

PROBLEM A

Random numbers

Input File:	A.IN
Output File:	STANDARD OUTPUT

It is given a random number generator implemented in the *C* programming language.

```
unsigned int m_w = 11;
unsigned int m_z = 173;
unsigned int myrandom(unsigned int M) {
  m_z = 36969 * (m_z & 65535) + (m_z >> 16);
  m_w = 18000 * (m_w & 65535) + (m_w >> 16);
  return ((m_z << 16) + m_w) % M;
}
```

```
A & B – bitwise AND operator
A >>n – bitwise right shift operator
A <<n – bitwise left shift operator
A % M – modulo operator
A = B – assignment
```

Using the given random number generator generate N distinct points in the plane $\{(x, y) | x, y \in \mathrm{N}$ (natural numbers), $0 \le x < M$, $0 \le y < M\}$. The numbers are generated until the desired number of distinct points is reached. During the generation it is likely to generate the same point more than once, which is called repetition. Write out the number of repetitions.

Restrictions and refinements

- $0 < M \le 65\,535$
- $0 < N \le 10\,00\,000$
- $N < M \times M$
- A point in the plane is generated by two consecutive call of `myrandom()` function.
  - `unsigned int x = myrandom();`
  - `unsigned int y = myrandom();`
- For each test case you should reset the random number generator: `m_w = 11; m_z = 173;`

Input

The input consists of several test cases. The first line of the input file contains a strictly positive number, denoting the number of the test cases. For each test case, we have two strictly positive natural numbers in a line, the first one is M, the parameter for the `myrandom` function and the second one N, the number of points to be generated.

Output

For each test case, the output file will have a line containing an integer number: the number of repetitions.

Input: A.in
1 4 10

Output: Standard output
4

Explanation

There is one test case: $M = 4$, $N = 10$

1.	2.	3.		4.	5.	6.	7.		8.			9.	10.
(0,3)	(2,0)	(3,2)	**(3,2)**	(3,1)	(0,2)	(2,1)	(1,1)	**(2,0)**	(2,2)	**(2,1)**	**(0,3)**	(2,3)	(1,0)

Result: 4, there are 4 repetitions (highlighted)

PROBLEM B

Detecting DOS attacks

Input File:	**B.IN**
Output File:	**STANDARD OUTPUT**

Our web programmer finished his newest web application and now he wants to protect it against DOS (denial of service) attacks. The web application records, for every request, the IP address of the remote computer and the reception time (seconds elapsed from the startup time of the web application). He should decide, for each computer, if it is attacker or not. If two distinct requests are coming from the same IP address, and the time difference between them is zero or one, that computer is declared as attacker.

Input

The input file stores the records in this format: the IP address (four numbers between 1 and 255, separated by dots), and the request time (integer value between 1 and 2 147 483 647). The records are stored in order of appearance, so the time-stamps appear in non-decreasing order.

Output

You have to read each record and, when a remote computer is identified as attacker for the first time, print its IP address.

Sample Input: B.IN
1.1.1.1 1 1.1.1.2 1 1.1.1.3 1 1.1.1.4 1 1.1.1.2 1 1.1.1.1 2 1.1.1.2 2 1.1.1.3 3 1.1.1.4 4

Sample Output: Standard output
1.1.1.2 1.1.1.1

Explanation

Examining the input list, we detect the first attacker: 1.1.1.2 (time-stamp 1), and the second attacker: 1.1.1.1 (time-stamp 2). The computer 1.1.1.2 is also recorded with time-stamp 2, but it is already banned so we ignore this record.

Restrictions and refinements

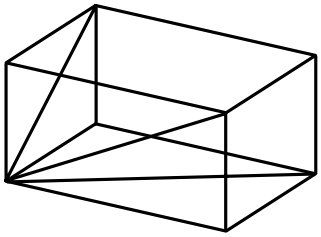
- The input file contains at most 480 000 records, and you have to manage at most 18 000 distinct IP addresses.
- Response time: two seconds.

PROBLEM C

LEGO Design

Input File:	C.IN
Output File:	STANDARD OUTPUT

The Lego Company is planning to start a new series of technique Lego having a better space flexibility. Currently every stick element fits with each other only in the three basic space directions: left-right, front-back and bottom-up. The first engineer received the challenge to design stick shape pieces which fit also diagonally in all the three basic space directions pairs, so that it would be able to build a parallelepiped, which has increased rigidity, containing not only the edges but also the three face-diagonals (see the figure below).



The first engineer quickly realized that the problem is – in mathematical terms – that all these six segments have to be commensurable, which in turn it means that all the three types of right-angled triangles has to have integer side length, forming Pythagorean triples. The smallest such triple is (3, 4, 5), as 3^2+4^2=5^2, and consequently using sticks of length 3, 4 and 5 not only they can form a right-angled triangle, but any two of them fit also "linearly": for example 5 pieces of sticks having length 3 fit exactly (it can be exchanged) with 3 pieces of sticks of length 5, both groups having the same total length.

The first mathematician of the company quickly listed all the Pythagorean triples (a, b, d) up to the greatest number 10 000, (a, b, $d < 10\,000$, a^2+b^2 = d^2) and supplied a text file containing a list of tab separated pairs (a, b), in each line, increasingly ordered by the first number, a. The third number d is useless, as certainly it is always computable knowing a and b, as d = sqrt(a^2+b^2) if needed.

Thereafter the first computer scientist of the company was giving the simple task to identify, using this list, all the triplets of such pairs, which fit together: more exactly the triplets (a, b, c), such that (a, b), (a, c), (b, c) are among the pairs of the given file.

More precisely, he has to list such triplets, avoiding repetitions by listing only the variant $a < b < c$, sorted increasingly by a, and he has to count the number of such reduced triplets. Can you help him?

Input

The input text file contains lines, each line having pairs of numbers (numbers not greater than 10 000, each number is followed by a tab character).

Output

The output file should contain either the text:

NO

...If no such triplets where found, or three lines:

- The first should contain: the number of triplets found
- The second one should contain the smallest (tab separated) triplet: *a b c*
- The third one should contain the greatest (tab separated) triplet: *a b c*

...If such triplets where found in the list.

Sample Input: C.IN

```
3     4
4     3
5     12
6     8
7     24
8     6
8     15
9     12
9     40
10    24
11    60
12    5
12    9
12    16
12    35
13    84
```

Sample Output: Standard output

(it reflects only the structure of file, not the correct values):

```
NO
```

or

```
1001
1     2     3
7777  8888  9999
```

PROBLEM D

Car Pooling

Input File:	D.IN
Output File:	STANDARD OUTPUT

As of the heavy snowing some of your friends have asked you to give them a ride tomorrow morning. What is the most efficient scheduling in terms of the total distance traveled?

For easier identification both you and your friends are identified through integer IDs. Your ID is always 0. Your friends' IDs are allocated in the order of request starting from 1. The pick-up/drop off points, referred as vertices from now on are also given integer IDs.

Input

The input file contains several problem specifications. These are separated by empty lines.

Each problem specifications starts with a positive integer, representing the number of seats in your car. Afterwards come lines of form *v*->*w*. The first of these specifies your original journey, you wanted to go from vertex *v* to vertex *w*. The next lines of the same format specify your friends' request. Then it comes a square matrix of size $n \times n$ where n is the total number of vertices. The elements of each row are separated by the space character. The (i, j)th element of this matrix gives you the length of the shortest route between vertex i and vertex j. All elements of this matrix are non-negative integers.

Output

Each solution is to be separated by an empty line. A solution is encoded as a sequence of actions which describe the state of the search, one action per line. Each line contains five elements separated by space character, encoded as follows:

fn vm pi {set1} {set2} OR
fn vm di {set1} {set2}
where
n is the total distance traveled so far
m is the current vertex
pi pick up person *i*
di drop off person *i*
set1 the set of indices of persons currently in the car
set2 the set of indices of the persons who haven't been picked up yet
In both *set1* and *set2* the elements are ordered ascendingly and separated by comma.

Remarks

- **Remark 1.** If in a given vertex it is possible to both pick up and to drop off people, perform drop off first.
- **Remark 2.** If you can pick up (drop off) more than one person at a time, pick up (drop off) people sorted ascendingly according to their IDs.
- **Remark 3.** Picking you up has to be the very first action. Dropping you down has to be the very last action.

Sample Input: D.IN

```
5
0->1
1->2
1->4
2->0
3->1
0 5 5 1 5
5 0 5 4 4
5 4 0 3 3
4 1 5 0 1
3 5 4 3 0

5
0->1
0->1
0->1
1->2
0 5 6
6 0 2
1 2 0
```

Sample Output: Standard output

```
f0 v0 p0 {0} {1,2,3,4}
f1 v3 p4 {0,4} {1,2,3}
f2 v1 d4 {0} {1,2,3}
f2 v1 p1 {0,1} {2,3}
f2 v1 p2 {0,1,2} {3}
f7 v2 d1 {0,2} {3}
f7 v2 p3 {0,2,3} {}
f10 v4 d2 {0,3} {}
f13 v0 d3 {0} {}
f18 v1 d0 {} {}

f0 v0 p0 {0} {1,2,3}
f0 v0 p1 {0,1} {2,3}
f0 v0 p2 {0,1,2} {3}
f5 v1 d1 {0,2} {3}
f5 v1 d2 {0} {3}
f5 v1 p3 {0,3} {}
f7 v2 d3 {0} {}
f9 v1 d0 {} {}
```

PROBLEM E

Evening meeting

Input File:	E.IN
Output File:	STANDARD OUTPUT

At an evening meeting some couple of people know each other, and some couple of people do not know each other. The acquaintance is considered mutual. If a person knows two other people who do not know each other, he presents them to each other in the evening.

Determine at the end of the night, how many people are in the most numerous group of people, which everyone knows everyone.

Input

Peoples are represented by natural numbers. The first line contains the number of tests. For each test the first line contains the number of people (< 100 000), the second line the number of acquainted couples, and the next lines the acquainted couples (two natural numbers).

Output

A resulted number for each test in a line.

Sample Input: E.IN
2 9 7 1 2 2 3 2 4 3 4 5 6 6 7 7 11 5 4 1 2 3 4 2 3 4 5

Sample Output: Standard output
4 5

PROBLEM F

Stargate

Input File:	F.IN
Output File:	STANDARD OUTPUT

In an imaginary universe, a stargate is a device that allows practical, rapid travel between two distant locations. There are a number of well-defined stargates in different positions of the universe and the connections between them are also well defined. The travel between two stargates is possible only if they are connected. During a journey even more stargates can be used to shorten the travel time. Our imaginary starship can reach a speed close to the speed of the light. For simplicity in our calculations, let's forget the theory of relativity and assume that:

- in the space, the starship will have a continuous speed c, where c is the speed of light
- there will be no obstacles on the way of the starship that should be bypassed
- the distance of a light year can be done in one year's time
- the travel between two connected stargates will require 0 time

Knowing the exact positions and connections between the stargates, calculate the minimal required travel time in years between the starting and ending point.

Input

The first line of the input file contains the number t of test cases. The first line of each test case represents the number s of stargates in the universe. The next s lines contains the coordinates of the stargates $G[i](x[i], y[i], z[i])$, measured from the center of the universe $O(0, 0, 0)$. The scale of the coordinate system is in light years. The upcoming line contains the number c of connections between stargates, followed by c lines each containing the ordinal number of two stargates connected bidirectionally. The numbering of the stargates starts at 1. The following two lines contain the coordinates of the starting $S(sx, sy, sz)$ and ending $E(ex, ey, ez)$ point of our journey.

Output

For each test case your program has to print on a new line the minimum traveling time in years between the starting and ending point in the universe, rounded to the nearest integer.

Constrains

- $1 \le t, s \le 100$
- $1 \le c \le 1000$
- $-10\,000 \le x[i], y[i], z[i], sx, sy, sz, ex, ey, ez \le 10\,000$
- every input and output value will be an integer

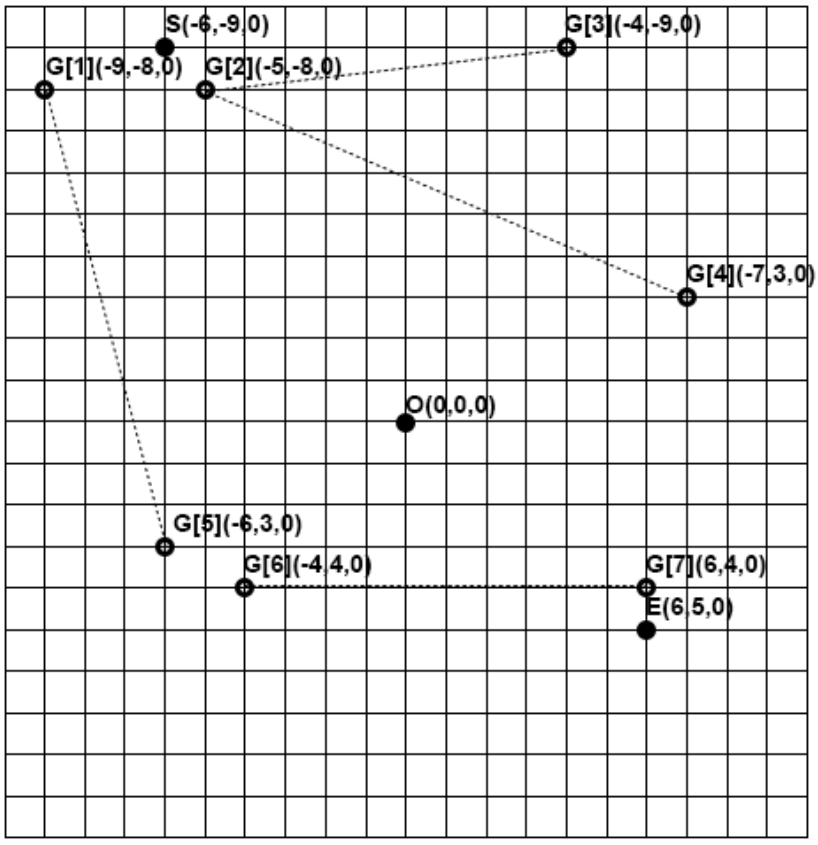
Sample Input: F.IN
1 8 -8 -8 0 -5 -8 0 -4 -9 0 -7 3 0 -6 3 0 -4 4 0 6 4 0 8 5 0 4 1 5 2 3 3 4 6 7 -6 -9 0 6 5 0

Sample Output: Standard output
6

Explanation

The following coordinate system represents the universe described in the example input file. The stargates are marked with small circles, while the center, starting and ending point with small filled dots. The dotted lines represent the connections between stargates. For simplicity, in our example, all the elements are placed on the *Z* = 0 plane.

The optimal route is: *S*, *G*[1], *G*[5], *G*[6], *G*[7], *E*, with a journey time of 3,16 + 0 + 2,24 + 0 + 1, which rounded to the nearest integer gives 6.



PROBLEM G

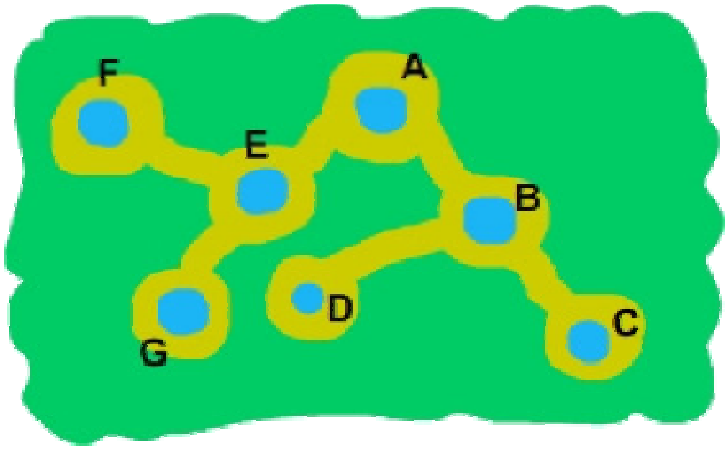
Wedding of Sultan[1]

Input File:	G.IN
Output File:	STANDARD OUTPUT

As usual, Sultan Mahmud is very busy. He works days and nights at the office. If you ask him, "Sultan, which day of the week is this?", he will look at you for a while and say, "I think I have 3 more days till deadline!" But one day, the scenario changed after receiving a call. He usually ignores phone calls from everyone (even from his fiancée), but this time he couldn't ignore it because of the importance of the person! This person was his to-be mother-in-law. So he answered the call and heard, "Son, only 30 days left until your wedding ceremony, so I am sending a tailor for the measurement of your suit." Sultan now remembered, he is about to get married but looking at himself, he got surprised! When did he get so fat! "Umm… Mom, can it be arranged 10 days later?" He wants to buy some time so that he can exercise and lose extra weight. So he immediately went out with his bicycle to the large garden beside his house.

There are several trails in the garden. A trail starts from one water sprinkler to another and the sprinkler are marked by distinct letters from "A" to "Z". The trails are designed in such a way that from the sprinkler at the entrance, you can go to any other sprinkler using exactly one path if you do not traverse a trail more than once.

While traversing the trails with his cycle, Sultan notes the names of the sprinklers in his notepad. He will write down the name of a sprinkler when he enters the sprinkler for the first time or leaves this sprinkler for the last time. And not surprisingly, geek Sultan follows a peculiar method to ensure that he visits all the trails of the garden. When he comes to a sprinkler, he looks for a trail which he has not traversed yet. If he finds such a trail, he follows that one. Otherwise, he uses the trail that he used to come here for the first time except if it's the entrance, when he stops exercising. He always starts from the entrance and, guess what, his peculiar strategy always guarantees him to finish at the entrance and all the trails to be visited.



For example, in the above garden the main entrance is at A. So Sultan will start from A. When Sultan is at A, he can choose either of the trails. Say he chooses the trail leading to E. Then he can choose the trail to G or the trail to F. Say he chooses F. Now he does not have any unvisited trail from F, so he will go back to E. Now he must choose the trail to G and then similarly will come back to E and back to A. Then he will go towards B. Now he again has two choices. He can go to C or D, say he goes to C, then he will be back to B, then will go to D, and hence back to B and also back to A, thus finishing his exercise. So after his exercise, you will find: AEFFGGEBDDCCBA in his notepad. Can you find the number of trails attached to the sprinklers just looking at the sequence written in the notepad?

1 http://uva.onlinejudge.org/external/125/12582.html, https://icpcarchive.ecs.baylor.edu/external/62/6201.html

Input

The first line of the input contains a positive integer T ($T \leq 100$), denoting the number of test cases.

Then T lines follow, each containing a valid sequence of sprinkler names. A sprinkler name will always be a capital Latin letter ("A", "B", …, "Z"). You may assume that there will be at least two sprinklers in the garden, otherwise there would be no meaning of exercise, right?

Output

For each case, output the case number in the first line, followed by the number of trails for each sprinkler. First print the sprinkler name, followed by the count of trails. These lines should be in lexicographical order of sprinkler names. Note that you should not print a sprinkler which is not present in the garden. Look at the sample input/output for more specific format of input/output.

Sample Input: G.IN
2 AEFFGGEBDDCCBA ZAABBZ

Sample Output: Standard output
Case 1 A = 2 B = 3 C = 1 D = 1 E = 3 F = 1 G = 1 Case 2 A = 1 B = 1 Z = 2

PROBLEM H

The Robot

Input File:	**H.IN**
Output File:	**STANDARD OUTPUT**

A calculation performing robot has been developed by engineers to be able to work very quickly with functions, and the same function can contain any number of arguments.

Unfortunately, the robot does not know anything else, so even a simple calculator can be realized by the robot as a sequence of function calls.

Write a program which converts a conventional arithmetic expression (that contains brackets "(" and ")", constant numbers, addition "+", subtraction "-", multiplication "*", division "/", exponentiation "^ ", positive and negative signs "+"and "-") into nested function calls.

If an arithmetical operation occurs one after the other several times, the corresponding function will have same number of arguments, as the number of constants carried out the operation.

Denote *add* the function for addition, *sub* the function for subtraction, *mul* – multiplication, *div* – division, *pow* - exponentiation, respectively *plus* and *minus* the functions for positive and negative signs.

Input

The first line of the input contains a positive integer T ($T \leq 100$), denoting the number of test cases.

Then T lines follow, each containing a valid (syntactically correct) arithmetic expression (string – not longer than 1000 characters).

Output

For each test case, write out a line (string), containing the arithmetical expression transcript to function calls.

Sample Input: H.IN
2 (3+4+5+6)/2 (1+2+3)*((7-4--2)^(3++2))

Sample Output: Standard output
div(add(3, 4, 5, 6), 2) mul(add(1, 2, 3), pow(sub(7, 4, minus(2)), add(3, plus(2))))

PROBLEM I

Decryptions

Input File:	I.IN
Output File:	STANDARD OUTPUT

A very simple, but insecure encryption method is to multiply an invertible 3×3 key matrix by each 3 bytes block of plain text again modulus 256 (determine the remainder of division by 256). Formally the encryption of 3 bytes plain text is the following:

$$Enc_K(M) : C = K \cdot M, \text{ where } M = (m_1, m_2, m_3),\ C = (c_1, c_2, c_3),$$

$$K = \begin{pmatrix} k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,1} & k_{2,2} & k_{2,3} \\ k_{3,1} & k_{3,2} & k_{3,3} \end{pmatrix}, \text{ and } \quad (c_1, c_2, c_3) = \begin{pmatrix} k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,1} & k_{2,2} & k_{2,3} \\ k_{3,1} & k_{3,2} & k_{3,3} \end{pmatrix} \cdot \begin{pmatrix} m_1 \\ m_2 \\ m_3 \end{pmatrix} \pmod{256}$$

In order to decrypt the ciphertext we multiply the inverse matrix of the key matrix by each 3 bytes block of the ciphertext. Encryption and decryption process cannot be done if the inverse matrix of key matrix again modulus 256 does not exist.

Write a program that for several encrypted byte sequence determines the plain text.

Input

The I.IN input file contains several test cases. The first single positive integers indicates the number of test cases followed by a blank line. There will be also a blank line between each two consecutive cases.

Each case consist of several lines:

- the first line contain the 3×3 key matrix,
- the second line contain a single positive integers that indicates the number of encrypted bytes,
- the third line contain the encrypted bytes.

Output

For each test cases try to decrypt the corresponding byte sequence. If decryption does not possible writes on the standard output "cannot decrypt" else writes the plain text obtained from decryption. After each plaintext you must to put a blank line.

Sample Input: I.IN

```
3

12 33 145 111 12 33 145 111 113
57
220 169 104 28 191 77 171 70 104 124 142 186 103 215 10 196 97 136 152 63
229 222 59 167 13 85 253 235 253 90 70 135 208 63 113 210 204 124 161 205
129 201 29 182 94 139 230 30 119 27 132 17 192 95 236 23 137

211 31 176 189 212 31 176 189 127
51
40 43 61 102 123 83 3 87 100 130 122 30 42 198 240 55 143 108 165 177 16
46 191 7 117 74 143 198 230 84 246 57 25 227 159 36 47 100 107 134 51 215
74 181 149 231 53 94 124 4 64

34 211 51 178 34 211 51 179 51
81
227 98 224 9 182 8 254 171 15 251 83 141 115 138 127 27 168 180 63 183 243
40 197 232 222 58 207 48 26 207 227 98 224 231 212 230 178 195 69 83 159
33 181 220 181 148 96 52 180 36 105 208 238 239 160 227 232 48 114 37 155
42 13 227 19 3 32 12 50 48 26 207 24 153 170 119 153 40 44 238 221
```

Sample Output: Standard output

```
Faculty of Technical And
Human Sciences,
Marosvasarhely

cannot decrypt

Marosvasarhely is the seat
of Maros County in the
north-central part of
Romania.
```

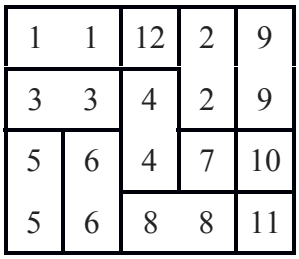
PROBLEM J

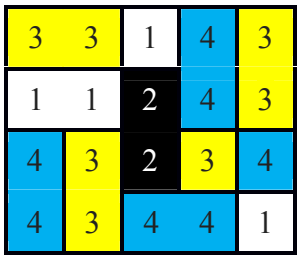
Room

Input File:	J.IN
Output File:	STANDARD OUTPUT

In this problem, you are in charge with painting the floor of a rectangular room according to some specific rules. The floor is already paved with 1×1 and 1×2 tiles, each of them labeled with a uniqe integer in {1, 2 , 3... *C*} where *C* is the total number of tiles. You must paint each individual tile such that no two tiles that share an edge have the same color. Also, you should do this using at most four distinct colors.

The example describes a 4×5 room which is paved using 12 tiles. A possible solution is illustrated on the right.





Task: Given the descriptions of several paved rooms, you should paint each one of them according to the aforementioned rules.

Input

The first line of the input file, J.IN, contains a single integer *T*. The following lines contain *T* test cases. Each individual test case contains the values *M* and *N* on its first line, separated by an empty space, representing the size of the room. The next *M* lines each contain *N* positive integers separated by **one or more** empty spaces. These integers describe the floor pavement.

Output

In the standard output, you should output the answers for each room, each tile being painted according to the task. Integers written on the same line should be separated by an empty space. Two consecutive test cases should be separated by an empty line.

Constraints

- $2 \leq M, N \leq 1000$,
- $2 \leq T \leq 10$,
- Any valid painting is accepted.
- A valid painting may use 4 colors or less.
- Time limit: 5 seconds

Sample Input: J.IN

```
2
4 5
1 1 12 2  9
3 3  4 2  9
5 6  4 7 10
5 6  8 8 11
3 3
1 1 2
3 4 2
3 5 5
```

Sample Output: Standard output

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

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

PROBLEM K

Sapientia-ECN 2030

Input File:	K.IN
Output File:	STANDARD OUTPUT

At the 25th edition of the Sapientia-ECN International Programming Contest participated *N* students (the student-IDs are: 1, 2, …, *N*) from *M* countries (the country-IDs are: 1, 2,…, *M*). Who was/were the most popular „international contestant(s)"?

We define the most popular „international contestant" as follows:

- 1st criteria: She/he has friends from the largest number of different countries.
- 2nd criteria: She/he has the largest number of foreign friends.
- 3rd criteria: She/he has the largest number of friends.

The selection procedure is described below:

- Firstly, you have to apply criteria 1;
  - In case of equality, you have to apply criteria 2;
    - In case of equality, you have to apply criteria 3.

Input

- The first row of the input file contains values *N* and *M* (separated by one space-character).
- The second row contains *N* natural numbers representing the country-ID of the corresponding students.
- Each next row contains a couple of student-ID (separated by one space-character) representing friendship relations.

Output

- The output file has to contain the ID(s) of the most popular „international contestant(s)"

Sample Input: K.IN
9 3 1 1 1 2 2 2 3 3 3 1 2 1 3 1 4 1 7 2 4 2 5 2 7

Sample Output: Standard output
2

Explanation

Both student-1 and student-2 has friends from 3 countries, but student-2 has 3 foreign friends and student-1 only 2.