Greetings From Globussoft

- Given below are 5 Programming questions, you have to solve any 3 out of 5 questions.

- These 5 questions you can attempt in any technology like C/C++, java, .Net, PHP

- To solve these 3 questions you've max. 3 hours.

- While Solving these questions you are not allowed to use any **Search Engine** like Google, Yahoo, Bing …

All the best for your test

Globussoft

QUESTION – 1

One of the traps we will encounter in the Pyramid is located in the Large Room. A lot of small holes are drilled into the floor. They look completely harmless at the first sight. But when activated, they start to throw out very hot java, uh ... pardon, lava. Unfortunately, all known paths to the Center Room (where the Sarcophagus is) contain a trigger that activates the trap. The ACM were not able to avoid that. But they have carefully monitored the positions of all the holes. So it is important to find the place in the Large Room that has the maximal distance from all the holes. This place is the safest in the entire room and the archaeologist has to hide there.

Input

The input consists of T test cases. The number of them (T) is given on the first line of the input file. Each test case begins with a line containing three integers X, Y, M separated by space. The numbers satisfy conditions: $1 <= X,Y <=10000$, $1 <= M <= 1000$. The numbers X and Yindicate the dimensions of the Large Room which has a rectangular shape. The number M stands for the number of holes. Then exactly M lines follow, each containing two integer numbers U_i and V_i ($0 <= U_i <= X$, $0 <= V_i <= Y$) indicating the coordinates of one hole. There may be several holes at the same position.

Output

Print exactly one line for each test case. The line should contain the sentence "`The safest point is (`*P*`,` *Q*`).`" where P and $Q$are the coordinates of the point in the room that has the maximum distance from the nearest hole, rounded to the nearest number with exactly one digit after the decimal point (0.05 rounds up to 0.1).

Example

```
Sample Input:
3
1000 50 1
10 10
100 100 4
10 10
10 90
90 10
90 90
3000 3000 4
1200 85
63 2500
2700 2650
2990 100

Sample output:
The safest point is (1000.0, 50.0).
The safest point is (50.0, 50.0).
The safest point is (1433.0, 1669.8).
```

QUESTION – 2

Agness, a student of computer science, is very keen on crosscountry running, and she participates in races organised every Saturday in a big park. Each of the participants obtains a route card, which specifies a sequence of checkpoints, which they need to visit in the given order. Agness is a very atractive girl, and a number of male runners have asked her for a date. She would like to choose one of them during the race. Thus she invited all her admirers to the park on Saturday and let the race decide. The winner would be the one, who scores the maximum number of points. Agnes came up with the following rules:

- a runner scores one point if he meets Agnes at the checkpoint,
- if a runner scored a point at the checkpoint, then he cannot get another point unless he and Agnes move to the next checkpoints specified in their cards.
- route specified by the card may cross the same checkpoint more than once,
- each competitor must strictly follow race instructions written on his card.

Between two consecutive meetings, the girl and the competitors may visit any number of checkpoints. The boys will be really doing their best, so you may assume, that each of them will be able to visit any number of checkpoints whilst Agnes runs between two consecutive ones on her route.

Task

Write a program which for each data set from a sequence of several data sets:

- reads in the contents of Agnes' race card and contents of race cards presented to Tom,
- computes the greatest number of times Tom is able to meet Agnes during the race,
- writes it to output.

Input

There is one integer d in the first line of the input file, 1 <= d <= 10. This is the number of data sets. The data sets follow. Each data set consists of a number of lines, with the first one specifying the route in Agnes' race card. Consecutive lines contain routes on cards presented to Tom. At least one route is presented to Tom. The route is given as a sequence of integers from interval [1, 1000] separated by single spaces. Number 0 stands for the end of the route, though when it is placed at the beginning of the line it means the end of data set. There are at least two and at most 2000 checkpoints in a race card.

Output

The i-th line of the output file should contain one integer. That integer should equal the greatest number of times Tom is able to meet with Agnes for race cards given in the i-th data set.

Example

```
Sample input:

3
1 2 3 4 5 6 7 8 9 0
1 3 8 2 0
2 5 7 8 9 0
1 1 1 1 1 1 2 3 0
1 3 1 3 5 7 8 9 3 4 0
1 2 35 0
0
1 3 5 7 0
3 7 5 1 0
0
1 2 1 1 0
1 1 1 0
0

Sample output:

6
2
3
```

QUESTION – 3

There are infinitely many coin denominations in the Byteland. They have values of 2^i for i=0,1,2,... . We will say that set of coins c1,c2,...,ck is perfect when it is possible to pay every amount of money between 0 and c1+...+ck using some of them (so {4,2,2,1} is perfect while {8,1} is not). The question is - is it always possible to change given sum n into a perfect set of coins? Of course it is possible ;). Your task will be more complicated: for a sum n you should find minimal number of coins in its perfect representation.

Input

First line of input contains one integer c<=50 - number of test cases. Then c lines follow, each of them consisting of exactly one integer n<=10^1000.

Output

For each test case output minimal number of coins.

Example

```
Input:
5
507
29
8574
```

```
233
149

Output:
14
7
21
11
10
```

QUESTION – 4

You are given two short sequences of numbers, X and Y. Try to determine the minimum number of steps of transformation required to convert sequence X into sequence Y, or determine that such a conversion is impossible.

In every step of transformation of a sequence, you are allowed to replace exactly one occerunce of one of its elements by a sequence of 2 or 3 numbers inserted in its place, according to a rule specified in the input file.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

For each test case, the first line of input contains four integers - N, M, U, V (1<=N,M<=50). The next two lines of input contain sequences X and Y, consisting of N and M integers respectively. The next U lines contain three integers: *a b c* each, signifying that integer *a* can be converted to the sequence *b c* in one step of transformation. The next V-U lines contain four integers: *a b c d* each, signifying that integer *a* can be converted to the sequence *b c d* in one step of transformation. With the exception of N and M, all integers provided at input are positive and do not exceed 30.

The format of one set of input data is illustrated below.

$N\ M\ U\ V$
$x_1\ x_2\ \ldots\ x_N$
$y_1\ y_2\ \ldots\ y_M$
$a_1\ b_1\ c_1$
$\vdots$
$a_U\ b_U\ c_U$
$a_{U+1}\ b_{U+1}\ c_{U+1}\ d_{U+1}$
$\vdots$
$a_V\ b_V\ c_V\ d_V$

Output

For each test case output -1 if it is impossible to convert sequence X into sequence Y, or the minimum number of steps required to achieve this conversion otherwise.

Example

Sample input:
```
1
3 10 2 3
2 3 1
2 1 1 2 2 1 2 1 2 1
3 1 2
3 3 3
3 1 3 2
```

Sample output:
```
6
```

QUESTION – 5

A diskette was enclosed to a road map. The diskette contains the table of the shortest ways (distances) between each pair of towns on the map. All the roads are two-way. The location of towns on the map has the following interesting property: *if the length of the shortest way from town A to town B equals the sum of the lengths of the shortest ways from A to C and C to B then town C lies on (certain) shortest way from A to B*. We say that towns A and B are neighbouring towns if there is no town C such that the length of the shortest way from A to B equals the sum of the lengths of the shortest ways from A to C and C to B. Find all the pairs of neighbouring towns.

Example

For the table of distances:

	A	**B**	**C**
A	0	1	2
B	1	0	3
C	2	3	0

the neighbouring towns are A, B and A, C.

Task

Write a program that for each test case:

- reads the table of distances from standard input;
- finds all the pairs of neighbouring towns;
- writes the result to standard output.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line.

In the first line of each test case there is an integer n, $1 <= n <= 200$, which equals the number of towns on the map. Towns are numbered from 1 to n.

The table of distances is written in the following n lines. In the $(i+1)$-th line, $1 <= i <= n$, there are n non-negative integers not greater than 200, separated by single spaces. The j-th integer is the distance between towns i and j.

Output

For each test case your program should write all the pairs of the neighbouring towns (i.e. their numbers). There should be one pair in each line. Each pair can appear only once. The numbers in each pair should be given in increasing order. Pairs should be ordered so that if the pair (a, b) precedes the pair (c, d) then $a < c$ or $(a = c$ and $b < d)$.

Consequent test cases should by separated by an empty line.

Example

Sample input:

```
1
3
0 1 2
1 0 3
2 3 0
```

Sample output:

```
1 2
1 3
```