International Collegiate Programming Contest
ACM-ICPC South Western European Regional
SWERC 2008

November, 23 2008

Problem Overview:

Good luck and have fun!

Problem A
Bring Your Own Horse

One of the essential activities of a knight is to compete in tournaments. Frequently, groups of knights gather around the country to compare their skills. On such a typical contest day, everyone has five hours to do ten disciplines, such as sword fighting, bow and arrow, and various forms of horseback riding. Needless to say, you have to bring your own horse.
This is not as easy as it seems. It often takes a knight several days to go from the castle where he lives to the place where a tournament is held. But horses sometimes are very, very stubborn. After having covered a certain distance on a single day, they sometimes simply stop and refuse to go any further. Luckily, they start anew on the next day. To make sure that the horse does not refuse service before the scheduled day trip is completed, a knight wants to choose an itinerary on which the longest day trip is as short as possible. Hence, a trip that takes many days with short distances is preferable over a shorter route that has the risk of a refusing horse.

Write a program which answers queries from knights spread all over the country about the best way to go from their castles to a tournament site. Given a description of the relevant places (i.e. castles, locations of tournaments, and hostels for overnight stays), the program should tell them the largest distance they have to cover on a single day so that this distance is minimal among all possible itineraries.
The places are designated by consecutive integers from 1 to N, while a road is represented by three integers, namely its place of origin, its destination, and its length. Every road can be used in both directions, and there is at least one route (i.e. a sequence of roads) between any two places. The knights stick to the given roads while travelling and spend their nights only at one of the N places.

Input

The first line contains the total number of test cases that follow.
Each test case begins with a line that first holds the number N of places ($1 \leq N \leq 3000$) followed by the number R of roads ($1 \leq R \leq 100000$). Then there are R lines with three integers each (a, b, and l), each of which defines a road connecting the places a and b ($1 \leq a, b \leq N$) with length l ($1 \leq l \leq 1000000$).
Thereafter, each test case continues with the number Q of queries on a line by itself ($1 \leq Q \leq 1000$). Each of the next Q lines holds two integers k and t, indicating a query by a knight who lives at place k and needs to go to a tournament at place t ($1 \leq k, t \leq N$, $k \neq t$).

Output

For each test case output a line containing the word "`Case`", a single space, and its serial number (starting with 1 for the first test case). Then, print one line for each query in this test case, containing the smallest maximal day trip distance as described above. Print a blank line after each test case.

(Sample Input and Output are provided on the next page)

Sample Input

```
2
4 4
1 2 100
2 3 100
3 4 100
4 1 200
1
1 4
6 9
2 4 5
5 1 7
3 6 6
3 1 4
2 3 2
1 2 1
6 5 42
4 5 3
4 6 5
4
1 3
3 4
5 4
6 1
```

Sample Output

```
Case 1
100

Case 2
2
5
3
5
```

Problem B
First Knight

Archibald and Adalbert are inseparable friends and the best knights of the whole kingdom. Competitive as they are, they occasionally engage in a little open-air sword fight, just to determine who among them really is the first knight. Of course, neither Adalbert nor Archibald wins, but they keep themselves busy for quite a while and get around in the surroundings. You have to calculate about how long their next 'fight' will last.

All the action takes place in a rectangular area which, for the sake of simplicity, is divided into unit squares numbered from $(1, 1)$ to (m, n). Starting at $(1, 1)$, the knights move from one square to one of the at most four adjacent squares, and finish as soon as they reach (m, n) where the tavern is located. At each square, it is largely a matter of chance where the fight will continue, but it also depends on the environment (for example, if a certain direction is uphill). Our model uses probabilities to decide into which adjacent square the fight will move next. (For example, an uphill direction has a lower probability.) It is your job to calculate the expected number of moves that are needed before the tavern is reached. You can assume that every move is independent of the directions chosen in the previous moves.

Input

The input consists of a sequence of rectangular areas. Each area starts with a line containing the dimensions of the rectangle m and n, where $1 \leq m, n \leq 40$. Four blocks follow that state the probability of a move in each direction. Every block contains m lines, and each line contains n numbers $p_{i,j}^{(k)}$, where $0 \leq p_{i,j}^{(k)} \leq 1$ for all $1 \leq i \leq m$ and $1 \leq j \leq n$ and $1 \leq k \leq 4$. The probabilities in block k are arranged as follows:

$$\begin{array}{cccccc} p_{1,1}^{(k)} & p_{1,2}^{(k)} & p_{1,3}^{(k)} & \cdots & p_{1,n-1}^{(k)} & p_{1,n}^{(k)} \\ p_{2,1}^{(k)} & p_{2,2}^{(k)} & p_{2,3}^{(k)} & \cdots & p_{2,n-1}^{(k)} & p_{2,n}^{(k)} \\ \vdots & & & & & \\ p_{m,1}^{(k)} & p_{m,2}^{(k)} & p_{m,3}^{(k)} & \cdots & p_{m,n-1}^{(k)} & p_{m,n}^{(k)} \end{array}$$

The number $p_{i,j}^{(k)}$ gives the probability of a move from square (i, j) to the next square: In block 1 this is $(i{+}1, j)$, in block 2 it is $(i, j{+}1)$, in block 3 it is $(i{-}1, j)$ and in block 4 it is $(i, j{-}1)$. For each square (i, j) except the tavern (m, n), the probabilities $p_{i,j}^{(k)}$ add up to 1 and at least one of $p_{i,j}^{(1)}$ or $p_{i,j}^{(2)}$ is not 0. (This ensures that the tavern will finally be reached with probability 1.) You may assume that the probability of moving outside the rectangle is 0, as are $p_{m,n}^{(k)}$ for all k. The sequence of areas is followed by a line containing two zeros.

Output

For each area, output a line containing the expected number of moves from $(1, 1)$ to (m, n). This number must have an absolute error less than 0.1 compared to the exact answer that is always less than 1000000.

(Sample Input and Output are provided on the next page)

Sample Input

```
2 2
0.01 0.50
0.00 0.00
0.99 0.00
0.50 0.00
0.00 0.00
0.50 0.00
0.00 0.50
0.00 0.00
1 5
0.0 0.0 0.0 0.0 0.0
1.0 0.1 0.7 0.5 0.0
0.0 0.0 0.0 0.0 0.0
0.0 0.9 0.3 0.5 0.0
3 3
0.000001 0.0 1.0
0.0 1.0 1.0
0.0 0.0 0.0
0.999999 1.0 0.0
1.0 0.0 0.0
0.000001 0.000001 0.0
0.0 0.0 0.0
0.0 0.0 0.0
0.999999 0.0 0.0
0.0 0.0 0.0
0.0 0.0 0.0
0.0 0.999999 0.0
0 0
```

Sample Output

```
4.0
41.142857142857146
7.999994000002
```

Problem C
Postal Charges

We are in the year 1308. The burgrave of Nuremberg comes to the conclusion that the winding streets of the city are a mess. Thus, he plans to rebuild the whole city in a much larger space with rectangular streets that are parallel to his favourite coordinate system.

It is well known that craftsmen belonging to different guilds should better be living in separate quarters of the city. Of course not all guilds have the same *reputation* and the quarters for the guilds are to be ordered by reputation. The quarters should also be ordered by the *importance* for the economy of the city, just to be able to evacuate – in case of an enemy attack or a fire – more important people first. Note that *importance* is not the same as *reputation*: e.g. whereas people working in finance have about the lowest possible reputation, they are still important for the economy of the city. In contrast, clerics have a high reputation although the economy could do quite well without them.
The burgrave places the quarter of a guild with reputation i and importance j in the corresponding square with coordinates $[i, i+1[\times[j, j+1[$, where i and j are integers. No two guilds share the same pair (i, j).

The only problem is that the burgrave does not know how much to charge for the new postal service. The price of every letter should be the same for all connections within the city. To find a fair price that covers the transportation costs, the burgrave needs to know the average path length of all the mail. A survey has shown that all connections are equally likely with the following exception: Nobody would ever consider to write a letter to someone of a guild with lower or equal reputation or importance than his own. We call a path between two houses admissible if it is not excluded by this condition (see figure on the next page).

Input

The input consists of several test cases, separated from each other by a blank line. The first line of each test case holds an integer n, $2 \leq n \leq 100800$. Each of the following n lines contains the coordinates of one house as two decimal numbers $0 \leq x, y < 10$, separated by one space. x, y may have up to 8 decimal places. Coordinates of some houses may be identical: this corresponds to multi-family houses.

Output

Your program should print one line for each test case and this line should contain the average length of the admissible paths as a decimal number rounded to 8 digits after the decimal point. As the streets are rectangular and parallel to the coordinate axes, you have to use the Manhattan or l_1-distance for the length of an individual path, i.e. the length of the path $(x_1, y_1) \to (x_2, y_2)$ is $|x_1 - x_2| + |y_1 - y_2|$. It is guaranteed that there is at least one admissible path.

(Sample Input and Output are provided on the next page)

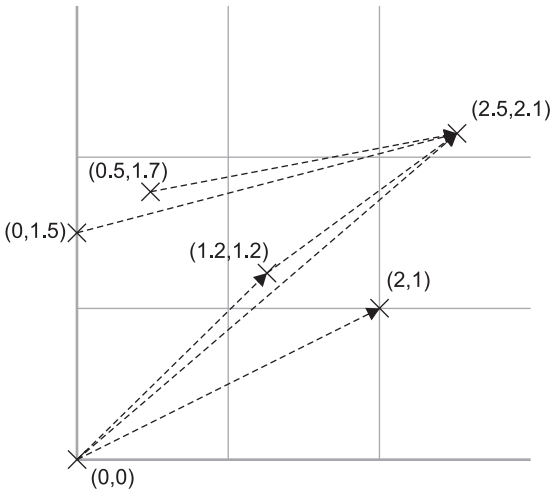

Figure 1 – The example city map (clipped) illustrates the last sample input, i.e. 9 quarters/guilds, 6 houses, and all admissible paths.

Sample Input

```
2
0 0
1 1

4
0 0
1.5 1.7
1.5 1.7
0 0

3
0.2 0.2
1.2 1.2
2.3 0.5

6
2 1
0 0
0 1.5
1.2 1.2
2.5 2.1
0.5 1.7
```

Sample Output

```
2.00000000
3.20000000
2.00000000
2.95000000
```

Problem D
Randomly-priced Tickets

At the end of the Middle Ages, quite a few universities throughout Europe have already been founded. The new term has just begun, so there are a lot of freshmen around. Not everyone has been lucky to be admitted to her/his desired university. As a result, many couples are now living in separate towns.

Of course, they try to see each other as often as they can. To facilitate this, the students have negotiated a deal with the coachmen. Instead of paying the regular price for a ride from one town to another, the price is determined by drawing a random integer between 1 and R inclusive, all numbers being equally likely. Unfortunately, this process repeats itself a few times whenever there is no direct connection between the towns a couple lives in. That makes the total cost of a journey quite unpredictable.

Help the couples determine the probability that one of them can afford a one-way trip to the other one. Given the number of towns and a list of direct connections, your program is supposed to process a list of couples. For each couple, you know their budget and where they live. Of course, they will always choose a route with the least expected price. Such a route exists between any two towns.

Input

The first line contains the number of test cases that follow.

Each test case begins with a line that holds the number N of towns ($1 \leq N \leq 100$) followed by the maximum price R of a single ticket ($1 \leq R \leq 30$). The following N lines contain N characters each. The j-th character in the i-th line of these is "`Y`" if there is a direct connection between towns i and j, but "`N`" otherwise. The j-th character in the i-th line is always the same as the the i-th character in the j-th line. The j-th character in the j-th line is always "`N`".

Each test case goes on with the number C of couples on a line by itself ($1 \leq C \leq 1000$). Then for each couple there is a line that holds three integers a, b, and m. These numbers state that one of them lives in town a, the other one in town b ($1 \leq a, b \leq N$, $a \neq b$), and the amount of money they can spend is m ($1 \leq m \leq 10000$).

Output

For each test case, print one line containing the word "`Case`", a single space, and its serial number (starting with 1 for the first test case). Then, output one line for each couple in this test case containing the probability that they can afford a one-way journey according to the rules above. Your answer is allowed to differ from the exact result by at most 0.001. Print a blank line after each test case.

(Sample Input and Output are provided on the next page)

Sample Input

```
2
3 4
NYY
YNY
YYN
1
1 3 1
4 7
NYNN
YNYN
NYNY
NNYN
2
1 3 10
1 4 10
```

Sample Output

```
Case 1
0.250000

Case 2
0.795918
0.341108
```

Problem E
The Game

As a touring merchant, I come to countries faraway. My wife allows me to travel around only if I return home with a nice, foreign gift in my hands. The *game*, which was my most recent present to her, was a really fascinating one.

The basic rules of the game are fairly simple: The game is played by exactly two players. In front of each player there is a fixed number of mugs and a bowl on the right side of the mugs. The mugs and bowls are arranged in a circle.

A number of stones is placed in each mug. The numbers of stones in the bowls are the scores of the corresponding players. The players alternate in making turns. In each turn, the current player selects one of his non-empty mugs. The stones are removed from the selected mug and are spread over the neighbouring mugs and bowls as follows: If there are N stones in the selected mug, one stone is added to each of the following N mugs/bowls in a counter-clockwise direction. Stones in the bowls count as scored points and cannot be removed in any further turn.

The following figure shows a turn in the mid of a game:

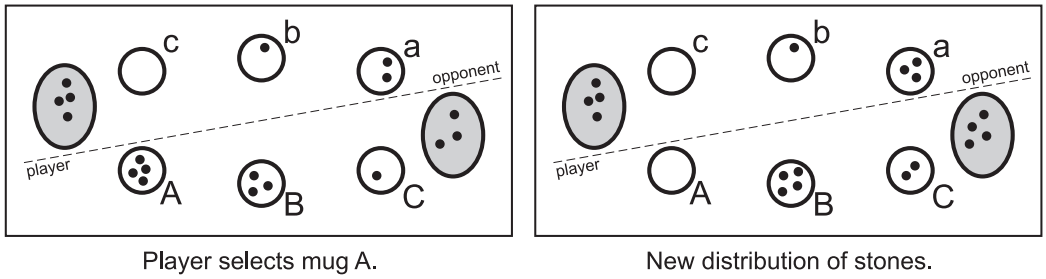

Figure 2 – Board with 3 mugs for each player, scores are 3:4 before the turn and 4:4 after afterwards.

If the last stone of a selected mug is added to the own *bowl*, the player gets an extra move. This extra move may result in additional extra moves (there is no limit for the number of extra moves).

If, on the other hand, the last stone is added to a mug of the opponent, the player has the option (not the obligation) to swap the opponent's mug with his *corresponding mug* (mugs A and a, mugs B and b, ...). Swapping is only allowed if the own mug is not empty. If he chooses to swap, then both mugs must not be swapped for the following four turns. (Note that this may be different from four *moves*: Consider the move sequence $P_0, O_1, O_2, O_3, P_4, O_5, P_6, O_7$ where P_i is a move by the player and O_j is a move by his opponent, who gets extra moves O_2 and O_3. If the player swaps mugs in move P_0, these two mugs may not be swapped again in moves O_1 to P_6, but again in O_7 and later moves.) Each player may choose the option to swap mugs up to three times within a game.

If the last stone is added to a mug of the current player, and if that mug was empty before distribution, and if the *opposite mug* of the opponent is not empty afterwards, the stones from both mugs are captured and put into the current player's bowl. This rule does not result in an extra move. Note that in general the *opposite* mug is different from the *corresponding* mug: Mugs A and c, mugs B and b, and mugs C and a are considered as opposite mugs in the above example.

The game ends as soon as every mug is empty and all stones are in the two bowls. If a player cannot move because all of his mugs are empty, but the opponent can move, it is the opponent's turn again. (With respect to the swapping rule above, the inactivity of the player who cannot move does count as a turn, too.) Otherwise, if a player can move, he has to choose one of the allowed moves.

Can you help me with the following question: What is the best difference between my final score and my opponent's final score that I can achieve from a given situation? In addition to me, my opponent also plays optimally.

Input

The first line gives the number of test cases T ($0 < T < 60$). Each test case is given in two lines. The first line of a test case holds the number of mugs M for each player ($0 < M < 5$). The second lines consists of $(M + 1) \cdot 2$ numbers, describing the current board after your previous turn.
The first M numbers describe the number of stones in my mugs in counter-clockwise order. The next number is my score (the stones in my bowl). Then follows the same for my opponent. You may assume that in total there are not more than 15 stones on the whole board.

Output

For each test case, print one line that gives the best difference between my score and the opponent's score at the end of the game. At first, it is the opponent's turn.

Sample Input

```
8
2
0 0 0 2 0 0
2
2 0 0 0 0 0
2
2 4 0 2 4 3
2
2 0 0 3 2 1
3
1 2 1 3 0 1 2 1
4
1 2 1 0 0 1 2 1 0 0
4
9 3 0 0 0 0 0 0 0 3
3
4 3 1 3 2 1 0 0
```

Sample Output

```
-2
2
-9
-4
5
0
1
-2
```

Problem F
The Merchant Guild

You are a warden of a small town's merchant guild. Early this morning, a number of local and foreign traders (numbered from 1 to n) line up in a row in order to enter the town's market lane. There are n locations along the lane where the merchants can place their carts and sell their goods. Beginning with the merchant #1, each merchant, one after the other, enters the lane with his cart, heads it to his assigned location, and, if it is free, occupies it. Otherwise he continues to the next free spot and occupies it. If all succeeding locations are occupied, he leaves without selling goods.

Traders are not able to turn their carts around because of the narrowness of the lane. Your job as a warden is to assign traders to locations in the lane. The local merchants are all members of the town's merchant guild and are privileged in that each of them gets assigned to his favourite location, whereas the foreign traders have to accept any spot you assign to them.

Given all local trader's desired locations, you have to decide whether there exists a valid assignment of foreign traders to locations in the lane such that every trader (both foreign and local) is able to find a free spot. If this is the case, you also have to find the number of different valid assignments modulo a given integer M.

Example: Assume there are four traders. The first three traders in the queue are local traders with favourite positions 2, 1 and 1 respectively. The last trader is a foreign one. Every merchant finds a free spot in the following four cases:

$$(2,1,1,1),\ \ (2,1,1,2),\ \ (2,1,1,3),\ \ (2,1,1,4)$$

where e.g. $(2,1,1,3)$ means that the first trader heads to position 2 first,..., and the last trader heads to position 3.
This example (which is the first test case of the sample input) shows that different local traders might have the same favourite location, that it is valid to assign a foreign trader to a spot that is desired by a local merchant and that a local merchant's final spot might not even be close to his favourite one.

Input

The first line contains the number of test cases. Each test case starts with a line with three integer numbers n, m, and M ($1 \leq n \leq 300, 0 \leq m \leq n, 2 \leq M \leq 10^9$), where m is the number of local merchants among all n traders. The next line contains exactly m pairs of integers $a_1, b_1, \ldots, a_m, b_m$ with $1 \leq a_i, b_i \leq n$ and all a_i different, where a_i is the position of the i-th local trader in the queue and b_i gives his favourite position. If there are no local traders, this line is empty.

Output

For each test case, output a single line, containing `NO`, if it is impossible that every merchant gets a free spot, or `YES` followed by the number of different assignments modulo M (separated by a single space).

(Sample Input and Output are provided on the next page.)

Sample Input

```
4
4 3 10
1 2 2 1 3 1
6 3 987654321
1 2 3 4 5 6
18 0 100769

10 3 8882
7 9 2 9 5 10
```

Sample Output

```
YES 4
YES 49
YES 68184
NO
```

Problem G
Toll Road

By an ingenious combination of warfare and arranged weddings, King Richard IV has gained control of a few remote areas of south-western Europe. (Actually, he is supposed to have coined the phrase that 'marriage is the continuation of war by different means'.) To profit from his new property, deputies shall be installed at certain roads to collect toll fees from passing travellers.

For each of the new areas, the royal cartographer has provided a simple map with the towns and major roads: Any two towns are connected by exactly one route (a sequence of roads from one town to another). To each road, the royal treasurer has assigned a number that indicates the profit he expects from collecting fees at that road. This number may be negative, which means that the cost of installing deputies is higher than the income.

Your task is to determine a *selection* of roads that maximizes the total profit (the sum of the earnings of all selected roads). It is not required that every town is at the end of a selected road, but the selection has to be connected: It must be possible to go from any selected road to any other selected road by using only selected roads (this makes transporting the collected fees safer).

Input

The input consists of a sequence of simple maps. Each map starts with a line containing the number of roads n, where $1 \leq n \leq 100000$. Each of the following n lines holds a road description that consists of three integer numbers a, b and p, where $1 \leq a, b \leq 200000$ and $-1000 \leq p \leq 1000$. They indicate the towns a and b at the ends of the road and the profit p of selecting this road. Towns are identified by unique numbers and roads may be passed in both directions. The sequence of maps is followed by a line containing a 0.

Output

For each map, output a line containing the maximum profit achievable by choosing a selection of roads as described above.

Sample Input

```
4
1 2 -7
3 2 10
2 4 2
5 4 -2
3
1 2 -8
2 3 -8
3 4 -1000
5
14 15 0
15 92 10
92 65 -5
65 35 10
35 89 -30
0
```

Sample Output

```
12
0
15
```

This page is intentionally left (almost) blank.

Problem H
Top Secret

Ralph was hired as a knight by a rich German earl some years ago. One day, he captured a spy that presumably intended to deliver a message to his earl's arch-enemy. At those ancient times, it was very popular to engrave a message into the inside of a ring in order to hide it. Knowing about this technique, Ralph quickly examined the spy's rings and found the message. But, unfortunately, it was encrypted.

Thus, Ralph tortured the spy until he disclosed how to decode the engraved message: The encrypted message is a single line of N numbers. One has to apply the following decoding procedure for S times: add to each number L times the number to its left and R times the number to its right. Note, that due to the cyclic engraving each number has exactly two neighbours. As numbers can be quite large, one only has to take care of the X lower digits.

Unfortunately, Ralph has never learnt how to add and multiply numbers. Please help him!

Input

The first line indicates the number T of test cases that follow. Test cases are separated by a blank line. Each test case starts with a line holding N, S, L, R, and X separated by single spaces ($2 < N \leq 1000$, $0 \leq S \leq 2^{30}$, $0 < L, R, X < 10$). The next line contains N numbers (separated by single spaces). These numbers are the encrypted message from left to right. Each of these numbers is a non-negative integer less than 1000.

Output

For each test case, output one line containing the N decrypted numbers, separated by single spaces.

Sample Input

```
5
3 1 1 1 3
1 1 1

3 0 1 1 3
23 42 0

3 1 1 1 3
23 42 0

4 10 2 1 9
1 2 3 4

5 999999999 3 8 7
8 7 8 7 12
```

Sample Output

```
3 3 3
23 42 0
65 65 65
2620960 2621920 2620896 2621984
2425139 2372828 6040064 4331745 9713040
```

This page is intentionally left (almost) blank.

Problem I
Transcribed Books

Long before Gutenberg invented letterpress printing, books have been transcribed by monks. Cloisters wanted to be able to check that a book was transcribed by them (and not by a different cloister). Although watermarked paper would have been an option, the cloister preferred to use a system of hard-to-fake serial numbers for identifying their transcriptions.
Each serial number consists of 10 single numbers $a_1, a_2, \ldots, a_{10}$. Valid serial numbers satisfy $a_1 + a_2 + \ldots + a_9 \equiv a_{10} \pmod{N}$ with $0 \leq a_{10} < N$. The N is specific to and only known by the cloister that has transcribed this book and is therefore able to check its origin.

You are confronted with a pile of books that presumably have been transcribed by a single cloister. You are asked to write a computer program to determine that cloister, i.e. to calculate the biggest possible N that makes the serial numbers of these books valid. Obviously, no cloister has chosen $N = 1$. So if your calculations yield $N = 1$, there must be something wrong.

Input

Input starts with an integer t on a single line, the number of test cases ($1 \leq t \leq 100$). Each test case starts with an integer c on a single line, the number of serial numbers you have to consider ($2 \leq c \leq 1000$). Each of the following c lines holds 10 integer numbers $a_1, a_2, \ldots, a_{10}$ ($0 \leq a_i < 2^{28}$) separated by single spaces.

Output

For each test case, output a single line containing the largest possible N, so that each given serial number for that test case is valid. If you cannot find a $N > 1$ satisfying the condition for all serial numbers or if the numbers are valid independent of the choice of N, output "`impossible`" (without the quotes) on a single line.

Sample Input

```
4
2
1 1 1 1 1 1 1 1 1 9
2 4 6 8 10 12 14 16 18 90
3
1 1 1 1 1 1 1 1 1 1
5 4 7 2 6 4 2 1 3 2
1 2 3 4 5 6 7 8 9 5
2
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 0
2
2 2 2 2 2 2 2 2 2 0
1 1 1 1 1 1 1 1 1 1
```

Sample Output

```
impossible
8
impossible
2
```

This page is intentionally left (almost) blank.

Problem J
Wizards

All through history, some people have been interested in the solutions of polynomial equations. As everybody knows, in the Middle Ages wizards were all around. They claimed to be able to find n solutions to any (univariate) polynomial equation of degree n. Of course, they sometimes needed to include some hocus-pocus like their magic number i, which they say is a solution to the equation $x^2 + 1 = 0$ (the second solution being $-i$).

But there are a few equations, for which most ordinary wizards failed to give n distinct solutions. Only the oldest and wisest wizards tried to be clever and bubbled something about multiplicity of roots – but nobody can possibly understand such excuses for finding fewer than n distinct roots.

Given a polynomial of degree n, find out if wizards can possibly find n distinct roots (including the magic ones using i), or if it is impossible — even for the wizards — to find n distinct roots.

Input

Input starts with the number of test cases t ($1 \leq t \leq 100$) in a single line. Each test case consists of a single line that holds a series of integers (separated by single spaces). The first integer is the degree n ($0 \leq n \leq 10$) of the polynomial in question. It is followed by the $n + 1$ coefficients $a_0 \ldots a_n$ ($-30 \leq a_i \leq 30$, $a_0 \neq 0$) to form the equation $\sum_{i=0}^{n} a_i x^{n-i} = 0$.

Output

For each test case output "`Yes!`" on a single line (without the quotes) if the wizards have a chance (provided they are as good as they claim) to find n distinct roots.
Print "`No!`" on a single line (again without quotes) if there is no way any wizard can possibly find n distinct roots.

Sample Input

```
5
2 1 1 1
2 1 2 1
4 1 2 1 2 1
4 1 2 2 2 1
4 1 0 2 0 1
```

Sample Output

```
Yes!
No!
Yes!
No!
No!
```

This page is intentionally left (almost) blank.