Problem C
The Stern-Brocot Number System

Input: standard input
Output: standard output

The *Stern-Brocot tree* is a beautiful way for constructing the set of all nonnegative fractions m / n where m and n are relatively prime. The idea is to start with two fractions $\left(\frac{0}{1}, \frac{1}{0}\right)$ and then repeat the following operations as many times as desired:

Insert $\frac{m+m'}{n+n'}$ between two adjacent fractions $\frac{m}{n}$ and $\frac{m'}{n'}$.

For example, the first step gives us one new entry between $\frac{0}{1}$ and $\frac{1}{0}$,

$$\frac{0}{1}, \frac{1}{1}, \frac{1}{0};$$

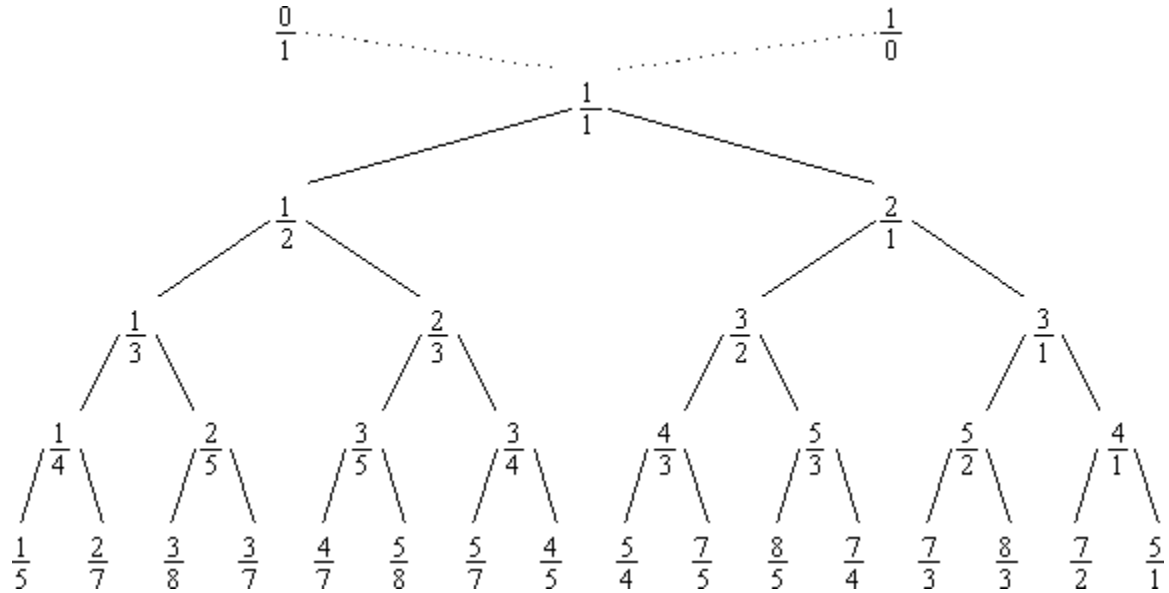
and the next gives two more:

$$\frac{0}{1}, \frac{1}{2}, \frac{1}{1}, \frac{2}{1}, \frac{1}{0}.$$

The next gives four more,

$$\frac{0}{1}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{1}{1}, \frac{3}{2}, \frac{2}{1}, \frac{3}{1}, \frac{1}{0};$$

and then we will get 8, 16, and so on. The entire array can be regarded as an infinite binary tree structure whose top levels look like this:



The construction preserves order, and we couldn't possibly get the same fraction in two different places.

We can, in fact, regard the *Stern-Brocot tree* as a *number system* for representing rational numbers, because each positive, reduced fraction occurs exactly once. Let's use the letters *L* and *R* to stand for going down to the left or right branch as we proceed from the root of the tree to a particular fraction; then a string of *L*'s and *R*'s uniquely identifies a place in the tree. For example, *LRRL* means that we go left from $\frac{1}{1}$ down to

$\frac{1}{2}$, then right to $\frac{2}{3}$, then right to $\frac{3}{4}$, then left to $\frac{5}{7}$. We can consider *LRRL* to be a representation of $\frac{5}{7}$. Every positive fraction gets represented in this way as a unique string of *L*'s and *R*'s.

Well, actually there's a slight problem: The fraction $\frac{1}{1}$ corresponds to the *empty* string, and we need a notation for that. Let's agree to call it *I*, because that looks something like 1 and it stands for "identity".

In this problem, given a positive rational fraction, you are expected to represent it in *Stern-Brocot number system*.

Input

The input file contains multiple test cases. Each test case consists of a line contains two positive integers *m* and *n* where *m* and *n* are relatively prime. The input terminates with a test case containing two 1's for *m* and *n*, and this case must not be processed.

Output

For each test case in the input file output a line containing the representation of the given fraction in the *Stern-Brocot number system*.

Sample Input

```
5 7
878 323
1 1
```

Sample Output

```
LRRL
RRLRRLRLLLLRLRRR
```

Rezaul Alam Chowdhury

Problem B: Savage Garden

Mr. Savage has a huge square-shaped garden named Savage Garden. The Savage Garden is very famous for its rare plants and amazing beauty. Currently, he is thinking of improving the garden by changing the orientation of plants into a special way. He wishes to divide the garden into smaller square regions such that the length of the complete square region is a power of two. Then, he wants to group the regions into ***L-shaped regions*** formed by three smaller squares. Each ***L-shaped region*** will have a different plant from the adjacent ***L-shaped regions***. He feels that the ***L*** will represent Love and his loved ones from all over the world and in the heavens can enjoy its beauty forever.

You have been hired by Mr. Savage to generate a plan for Savage Garden.

Input

Input consists of several test cases. Each test case specifies three integers, ***N***, ***X*** and ***Y***. ***N***, ***1 N 10*** represents that Savage Garden is of size 2^N ***X*** 2^N, after division. ***X*** and ***Y*** indicate the region where his house is located. For example, topmost-leftmost region is indicated by ***X=1*** and ***Y=1***. Of course, he doesn't want his house to be planted with saplings. So, make sure you leave his house intact while deriving the plan. You can assume that the co-ordinates given are always within the square region.

Output

For each test case, output a square matrix showing the Savage Garden. Indicate his house by the character '*'. All other regions should be indicated by alphabets 'a'-'z'. Character used to represent one ***L-shaped region*** should be different from the ones used to represent its neighboring ***L-shaped regions***. Assume that each small square region has 8 neighbors. If there are many solutions, you may output any of them. You may assume that there is at least one solution for the given test cases. Output a blank line after each test case.

Sample Input

```
2 3 4
```

Sample Output

```
aadd
abbd
cbee
cc*e
```

Arun Kishore

Problem C : Fill the Containers

Time Limit: 1 sec

A conveyor belt has a number of vessels of different capacities each filled to brim with milk. The milk from conveyor belt is to be filled into 'm' containers. The constraints are:

- Whenever milk from a vessel is poured into a container, the milk in the vessel must be completely poured into that container only. That is milk from same vessel can not be poured into different containers.
- The milk from the vessel must be poured into the container in order which they appear in the conveyor belt. That is, you cannot randomly pick up a vessel from the conveyor belt and fill the container.
- The ith container must be filled with milk only from those vessels that appear earlier to those that fill jth container, for all i < j

Given the number of containers 'm', you have to fill the containers with milk from all the vessels, without leaving any milk in the vessel. The containers need not necessarily have same capacity. You are given the liberty to assign any possible capacities to them. Your job is to find out the minimal possible capacity of the container which has maximal capacity. (If this sounds confusing, read down for more explanations.)

Input Format

A single test case consist of 2 lines. The first line specifies 1<=n<=1000 the number of vessels in the conveyor belt and then 'm' which specifies the number of containers to which, you have to transfer the milk. (1 <= m <= 1000000) .The next line contains, the capacity 1<=c<=1000000 of each vessel in order which they appear in the conveyor belt. Note that, milk is filled to the brim of any vessel. So the capacity of the vessel is equal to the amount of milk in it. There are several test cases terminated by EOF.

Output Format

For each test case, print the minimal possible capacity of the container with maximal capacity. That is there exists a maximal capacity of the containers, below which you can not fill the containers without increasing the number of containers. You have to find such capacity and print it on a single line.

Sample Input

5 3

1 2 3 4 5
3 2
4 78 9

Sample Output

6
82

Explanation of the output:

Here you are given 3 vessels at your disposal, for which you are free to assign the capacity. You can transfer, {1 2 3} to the first container, {4} to the second, {5} to third. Here the maximal capacity of the container is the first one which has a capacity of 6. Note that this is optimal too. That is, you can not have the maximal container, have a capacity, less than 6 and still use 3 containers only and fill the containers with all milk.

For the second one, the optimal way is, {4 78} into the first container, and {9} to the second container. So the minimal value of the maximal capacity is 82. Note that {4} to first container and {78 9} to the second is not optimal as, there exists a way to have an assignement of maximal capacity to 82, as opposed to 87 in this case.

Problem Setter: Rajesh S R
Written for CarteBlanche '08

Problem A
The Most Distant State

Input: standard input
Output: standard output

The 8-puzzle is a square tray in which eight square tiles are placed. The remaining ninth square is uncovered. Each tile has a number on it. A tile that is adjacent to the blank space can be slid into that space. A game consists of a starting state and a specified goal state. The starting state can be transformed into the goal state by sliding (moving) the tiles around. The 8-puzzle problem asks you to do the transformation in minimum number of moves.

2	**8**	**3**
1	**6**	**4**
7		**5**

Start

=>

1	**2**	**3**
8		**4**
7	**6**	**5**

Goal

However, our current problem is a bit different. In this problem, given an initial state of the puzzle your are asked to discover a goal state which is the most distant (in terms of number of moves) of all the states reachable from the given state.

Input

The first line of the input file contains an integer representing the number of test cases to follow. A blank line follows this line.

Each test case consists of 3 lines of 3 integers each representing the initial state of the puzzle. The blank space is represented by a 0 (zero). A blank line follows each test case.

Output

For each test case first output the puzzle number. The next 3 lines will contain 3 integers each representing one of the most distant states reachable from the given state. The next line will contain the shortest sequence of moves that will transform the given state to that state. The move is actually the movement of the blank space represented by four directions : U (Up), L (Left), D (Down) and R (Right). After each test case output an empty line.

Sample Input

```
1

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

Sample Output

```
P          u          z          z          l          e
8                     1                     5
7                     3                     6
4                     0                     2
U          U          R          D          D          R          U
```

Rezaul Alam Chowdhury

"*A fool looks for happiness in the distance, those who are intelligent grow it under their own feet.*"

Beautiful Points

Time limit: ? seconds
Memory limit: 64 megabytes

There are several points on the plane named *beauty points*. Given a point A, its *ugliness* is defined as |AB|+|AC|, where B and C are two beauty points nearest to A.

Your task is: given beauty points, find the most beautiful point, i.e., the point having least ugliness. Note: the most beautiful point doesn't have to be a beauty point.

Input

The first line of the input contains the number of the test cases, which is at most 10. The descriptions of the test cases follow. The first line of a test case descriptions contains an integer N ($2 \le N \le 10000$), which is the number of beauty points. Each of the next N lines contains two integers X and Y separated by a space ($-10000 \le X, Y \le 10000$) being the coordinates of a beauty point. No two beauty points in a test case description have the same coordinates. The test cases are separated by blank lines.

Output

For each test case in the input, output the coordinates of any most beautiful point separated by a space, with at least three digits after the decimal point. Print a blank line between test cases.

Examples

Input	Output
2 4 0 0 0 1 1 1 1 0 4 -1 -1 0 0 1 0 2 1	0.500 0.000 0.500 0.000

Problem E: Slalom

You are competing in a ski slalom, and you need to select the best skis for the race. The format of the race is that there are N pairs of left and right gates, where each right gate is W metres to the right of its corresponding left gate, and you may neither pass to the left of the left gate nor to the right of the right gate. The i^{th} pair of gates occurs at distance y_i down the hill, with the horizontal position of the i^{th} left gate given by x_i. Each gate is further down the hill than the previous gate (i.e. $y_i < y_{i+1}$ for all i).

You may select from S pairs of skis, where the j^{th} pair has speed s_j. Your movement is governed by the following rule: if you select a pair of skis with speed s_j, you move with a constant downward velocity of s_j metres per second. Additionally, at any time you may move at a horizontal speed of at most v_h metres per second.

You may start and finish at any two horizontal positions. Determine which pair of skis will allow you to get through the race course, passing through all the gates, in the shortest amount of time.

Input Specification

The first line of input contains a single integer, the number of test cases to follow.

The first line of each test case contains the three integers W, v_h, and N, separated by spaces, with $1 <= W <= 10^8$, $1 <= v_h <= 10^6$, and $1 <= N <= 10^5$.

The following N lines of the test case each contain two integers x_i and y_i, the horizontal and vertical positions respectively of the i^{th} left gate, with $1 <= x_i, y_i <= 10^8$.

The next line of the test case contains an integer S, the number of skis, with $1 <= S <= 10^6$.

The following S lines of the test case each contain one integer s_j, the speed of the j^{th} pair of skis, with $1 <= s_j <= 10^6$.

Sample Input

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

```
5 2
1 3
3
3
2
1
3 2 3
1 1
5 2
1 3
1
3
```

Output Specification

Output one line for each test case. If it is impossible to complete the race with any pair of skis, print the line `IMPOSSIBLE`. Otherwise, print the vertical speed s_j of the pair of skis that allows you to get through the race course in the shortest time.

Output for Sample Input

```
2
IMPOSSIBLE
```

Malcolm Sharpe, Ondřej Lhoták

A Marble Game is played with M marbles on a square board. The board is divided into $N \times N$ unit squares, and M of those unit squares contain holes. Marbles and holes are numbered from 1 to M . The goal of the Marble game is to roll each marble into the hole that has the same number.

A game board may contain walls. Each wall is one unit long and stands between two adjacent unit squares. Two squares are considered adjacent if and only if they share a side.

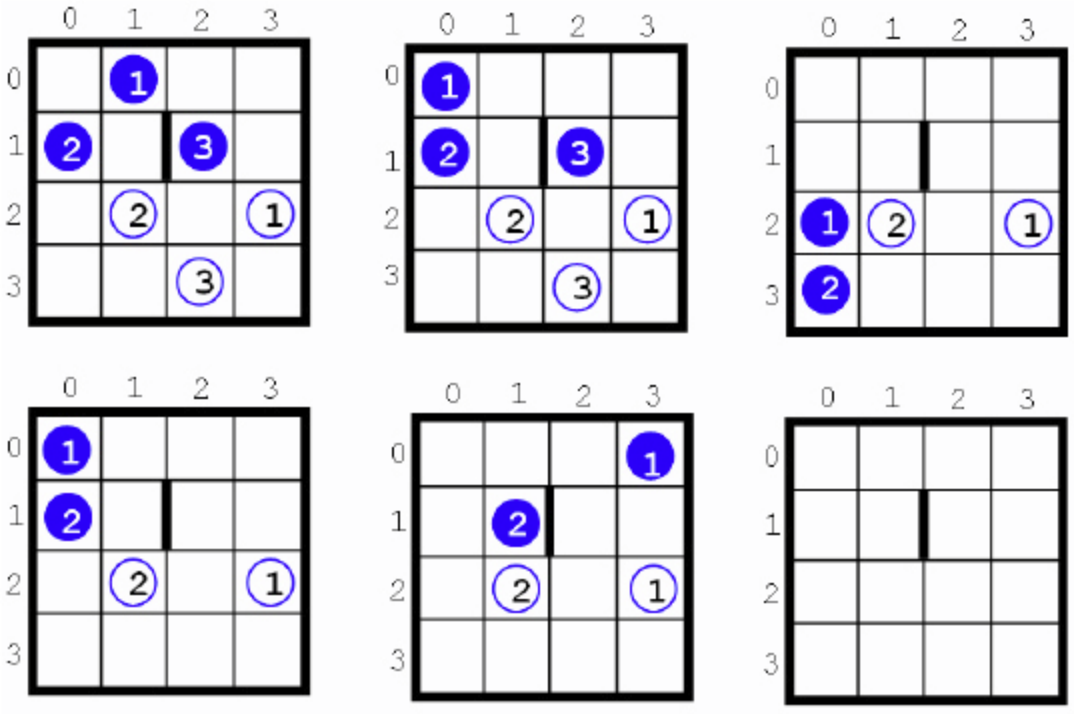
At the beginning of the game, all marbles are placed on the board, each in a different square. A ``move" consists of slightly lifting a side of the game board. Then all marbles on the board roll downward toward the opposite side, each one rolling until it meets a wall or drops into an empty hole, or until the next square is already occupied by another marble. Marbles roll subject to the following restrictions:

- Marbles cannot jump over walls, other marbles, or empty holes.
- Marbles cannot leave the board. (The edge of the board is a wall.)
- A unit square can contain at most a single marble at any one time.
- When a marble moves into a square with a hole, the marble drops into that hole. The hole is then filled, and other marbles can subsequently roll over the hole. A marble in a hole can never leave that hole.

The game is over when each marble has dropped into a hole with the corresponding number.

The figure below illustrates a solution for a game played on a 4×4 board with three blue marbles, three holes and a wall. The solution has five moves: lift the east side, lift the north side, lift the south side, lift the west side, lift the north side.

Your program should determine the fewest number of moves to drop all the marbles into the correct holes -- if such a move sequence is possible.



Input

The input file contains several test cases. The first line of each test case contains three numbers: the size N ($2 \leq N \leq 4$) of the board, the number M ($M > 0$) of marbles,

and the number W of walls. Each of the following $2M$ lines contains two integers. The first integer is a row location and the second is a column location. The first M of those lines represent the locations of the marbles, where marble#1 is on the first line, marble#2 on the second, and so on. The last M of those lines represent the locations of the holes, with the location of hole#1 coming first, hole#2 coming second, and so on. Finally, the next W lines represent the wall locations. Each of those lines contains four integers: the first pair are the row and column of the square on one side of the wall and the second pair are the row and column of the square on the other side of the wall. Rows and columns are numbered 0..N - 1 .

The input file ends with a line containing three zeroes.

Output

For each test case, print the case number (beginning with 1) and the minimal number of moves to win the game. If the game cannot be won, print the word ``impossible". Put a blank line after each test case. Use the format of the sample output below.

Sample Input

```
4 3 1
0 1
1 0
1 2
2 3
```

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

Sample Output

```
Case 1: 5 moves

Case 2: impossible
```

Problem D.Number Game

Background

Let’s play a number game. I will give you 2N-1(N=2^k, k=1,2,3,4,5,6,7,8,9,10) numbers, each number is a positive integer not bigger than 1000. Can you choose N of them, and add them all to a integer S, to make that S/N is a integer? If there are many solutions, you can only find one of them.

Input

The input file contains several scenarios. Each of them consists of 2 lines.

For each scenario, the first line is a number N, the second line consist of 2N-1 numbers. There is a space between two numbers.

The file will be ended with N=0.

Output

For each scenario, print a single line ‘No’ if you can’t find an answer. Otherwise print a line ‘Yes’, and then the other line containing N numbers (in any order), there should be a space between two numbers.

Sample Input

```
2
1 2 3
4
1 2 3 4 5 6 7
0
```

Sample Output

```
Yes
1 3
Yes
1 3 5 7
```