Problem A: Ideas

A unique feature of ideas is that they are not consumed when used. A good idea can benefit arbitrarily many people without diminishing the value of the idea. An idea can even serve as the basis for creative people to derive even better ideas. Each person relies on a specific set of ideas to build on and create new ideas.

In order to realize these benefits, ideas need to be communicated to the people who use them. People have developed an extensive worldwide communication network to satisfy this need. The network is composed of a series of tubes connecting people. The tubes are inhabited by curious creatures called packets which carry ideas from one person to another. In order to avoid collisions between packets, the tubes are all unidirectional. Each person can have zero or more incoming and zero or more outgoing tubes.

All of the packets start at the same person, and each packet follows the following algorithm:

1. Learn and remember the ideas created by the current person.
2. If there are no outgoing tubes from the current person, stop executing the algorithm.
3. Otherwise, choose an arbitrary outgoing tube and use it to travel to a new person.
4. Tell the new person all of the ideas that he or she needs from other people.
5. Go back to step 1.

The input data is such that it is possible for a packet to reach every person from person 0, and whenever a person P relies on a given idea, every path a packet could have taken to reach P will have visited at least one person who created that idea. It is possible for more than one person to independently create the same idea.

To ease the strain on each packet, you would like to minimize the number of ideas that it remembers at any given time by directing the packet to forget certain

ideas in certain tubes. However, in so doing, you must ensure that no matter what path the packet takes, every time it visits a person, the packet knows all of the ideas that the person needs.

Input Specification

The first line of input contains a single integer, the number of test cases to follow. Each test case begins with a line containing three integers N, M, I, the number of people, tubes, and ideas, respectively. Each of these integers is between 1 and one thousand, inclusive. People are numbered from 0 to N-1, ideas from 0 to I-1. All of the packets start at person 0. 2N lines follow, two lines for each person in order from 0 to N-1. Each of these lines contains some number of integers separated by spaces. For each person, the first line lists the ideas that the person needs, and the second line lists the ideas that the person creates. A given idea will never appear on both lines for the same person. M more lines follow, each describing a tube using two integers: the source and destination person of the tube.

Sample Input

```
1
3 3 2

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

Output Specification

For each test case, output M lines, each corresponding to one of the tubes in the same order as in the input. For each tube, output a list of integers: the minimal set of ideas the packet must have in its memory when travelling through the given tube. Output the ideas in each set in increasing order.

Output for Sample Input

```
1
0
1
```

Ondřej Lhoták

Problem B: Balance

A sailboat has many forces acting on it. This allows it to sail in many different directions, even if the wind is not blowing in the desired direction. In order for the boat to be easily controlled, however, certain forces must be balanced.

Suppose the wind is blowing from the north and the boat is facing west. Above the water, the blowing wind hits the sails, and because of the way that they are angled, pushes the boat to the southwest. The keel extends deep below the water. The water resistance on the keel creates a counterforce pushing the boat northward. Ideally, the northward force on the keel will counteract the southward component of the force on the sails, so that the boat will move to the west.

A problem can arise if the centre of the sails (called the Centre of Effort, or CE) is not directly above the centre of the keel (called the Centre of Lateral Resistance, or CLR). In general, the boat can pivot about the centre of the keel. If the sails are too far forward of the keel, the wind will push the bow (the front) of the boat southwards, and the boat will turn towards the south. If the sails are too far aft of (behind) the keel, the wind will push the stern (the back) southwards, and the boat will turn towards the north. Ideally, the sails and keel are balanced so that the boat sails in a straight line.

In this problem, you will examine a side view of the boat to determine whether the CE is above the CLR. The CE is defined as the centroid of the part of the boat above the waterline. The CLR is defined as the centroid of the part of the boat below the waterline. The centroid of a polygon is the unique point such that any line passing through it divides the polygon into two halves of equal area.

Input Specification

The first line will contian the number of test cases to follow. The first line of each case will contain a single integer n specifying the number of points along the outline of the side view of the boat. The following n lines will each contain two integers: the x and y coordinate of a point along the outline. The points will be given in order along the outline. The x axis (i.e. the line $y = 0$) represents the waterline. Assume that the boat faces in the direction of increasing x coordinates.

Sample Input

```
1
7
0 1
2 3
```

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

Output Specification

For each case:

If the CE is forward of the CLR, print the line:

```
CE is forward of CLR by N units.
```

If the CE is aft of (behind) the CLR, print the line:

```
CE is aft of CLR by N units.
```

In both cases, replace the `N` with the difference in the x coordinates of the CE and CLR to two decimal places. If the x coordinates of the CE and CLR are equal, print the line:

```
Balanced.
```

Output for Sample Input

```
CE is aft of CLR by 0.50 units.
```

Ondřej Lhoták

Problem C: Trainsorting

Erin is an engineer. She drives trains. She also arranges the cars within each train. She prefers to put the cars in decreasing order of weight, with the heaviest car at the front of the train.

Unfortunately, sorting train cars is not easy. One cannot simply pick up a car and place it somewhere else. It is impractical to insert a car within an existing train. A car may only be added to the beginning and end of the train.

Cars arrive at the train station in a predetermined order. When each car arrives, Erin can add it to the beginning or end of her train, or refuse to add it at all. The resulting train should be as long as possible, but the cars within it must be ordered by weight.

Given the weights of the cars in the order in which they arrive, what is the longest train that Erin can make?

Input Specification

The first line of input contains a single integer, the number of test cases to follow. The first line of each case contains an integer 0 <= n <= 2000, the number of cars. Each of the following n lines contains a non-negative integer giving the weight of a car. No two cars have the same weight.

Sample Input

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

Output Specification

For each case, output a line containing a single integer giving the number of cars in the longest train that can be made with the given restrictions.

Output for Sample Input

```
3
```

Problem D: Brownie Points II

Stan and Ollie play the game of Odd Brownie Points. Some brownie points are located in the plane, at integer coordinates. Stan plays first and places a vertical line in the plane. The line must go through a brownie point and may cross many (with the same x-coordinate). Then Ollie places a horizontal line that must cross a brownie point already crossed by the vertical line.

Those lines divide the plane into four quadrants. The quadrant containing points with arbitrarily large positive coordinates is the top-right quadrant.

The players score according to the number of brownie points in the quadrants. If a brownie point is crossed by a line, it doesn't count. Stan gets a point for each (uncrossed) brownie point in the top-right and bottom-left quadrants. Ollie gets a point for each (uncrossed) brownie point in the top-left and bottom-right quadrants.

Stan and Ollie each try to maximize his own score. When Stan plays, he considers the responses, and chooses a line which maximizes his smallest-possible score.

Input contains a number of test cases. The data of each test case appear on a sequence of input lines. The first line of each test case contains a positive odd integer $1 < n < 200000$ which is the number of brownie points. Each of the following n lines contains two integers, the horizontal (x) and vertical (y) coordinates of a brownie point. No two brownie points occupy the same place. The input ends with a line containing 0 (instead of the n of a test).

For each input test, print a line of output in the format shown below. The first number is the largest score which Stan can assure for himself. The remaining numbers are the possible (high) scores of Ollie, in increasing order.

Sample input

```
11
3 2
3 3
```

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

Output for sample input

```
Stan: 7; Ollie: 2 3;
```

P. Rudnicki

Problem E: Fire!

Joe works in a maze. Unfortunately, portions of the maze have caught on fire, and the owner of the maze neglected to create a fire escape plan. Help Joe escape the maze.

Given Joe's location in the maze and which squares of the maze are on fire, you must determine whether Joe can exit the maze before the fire reaches him, and how fast he can do it.

Joe and the fire each move one square per minute, vertically or horizontally (not diagonally). The fire spreads all four directions from each square that is on fire. Joe may exit the maze from any square that borders the edge of the maze. Neither Joe nor the fire may enter a square that is occupied by a wall.

Input Specification

The first line of input contains a single integer, the number of test cases to follow. The first line of each case contains the two integers R and C, separated by spaces, with $1 <= R,C <= 1000$. The following R lines of input each contain one row of the maze. Each of these lines contains exactly C characters, and each of these characters is one of:

- `#`, a wall
- `.`, a passable square
- `J`, Joe's initial position in the maze, which is a passable square
- `F`, a square that is on fire

Sample Input

```
2
4 4
####
#JF#
#..#
#..#
3 3
```

```
###
#J.
#.F
```

Output Specification

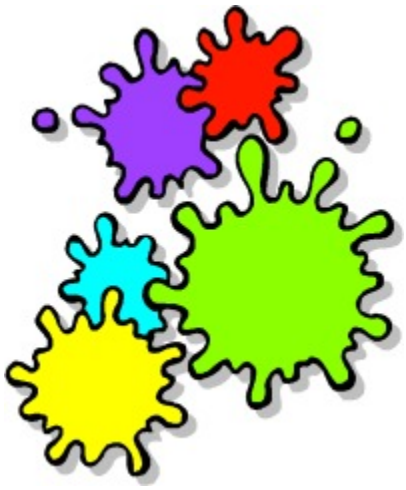
For each case, output a single line containing `IMPOSSIBLE` if Joe cannot exit the maze before the fire reaches him, or an integer giving the earliest time Joe can safely exit the maze, in minutes.

Output for Sample Input

```
3
IMPOSSIBLE
```

Malcolm Sharpe, Ondřej Lhoták

Problem F - Paintball



You are playing paintball on a 1000x1000 square field. A number of your opponents are on the field hiding behind trees at various positions. Each opponent can fire a paintball a certain distance in any direction. Can you cross the field without being hit by a paintball?

Assume that the southwest corner of the field is at (0,0) and the northwest corner at (0,1000). The first line of input contains a single integer, the number of test cases to follow. Each case consists of a line containing n <= 1000, the number of opponents. A line follows for each opponent, containing three real numbers: the (x,y) location of the opponent and its firing range. The opponent can hit you with a paintball if you ever pass within his firing range.

You must enter the field somewhere between the southwest and northwest corner and must leave somewhere between the southeast and northeast corners.

For each case, if you can complete the trip, output four real numbers with two digits after the decimal place, the coordinates at which you may enter and leave the field, separated by spaces. If you can enter and leave at several places, give the most northerly. If there is no such pair of positions, print the line:
`IMPOSSIBLE`

Sample Input

```
1
3
500 500 499
0 0 999
1000 1000 200
```

Output for Sample Input

```
0.00 1000.00 1000.00 800.00
```

Gordon Cormack

Problem G: Gangs

The downtown core of Inner City is laid out as a grid, with numbered streets running north-south from 1st Street in the west to 20th Street in the east, and numbered avenues running east-west from 1st Avenue in the north to 20th Avenue in the south. The area is controlled by two gangs, the Blips and the Cruds. The boundary between their territory is the Green Line, running diagonally from the intersection of 1st Street and 1st Avenue to the intersection of 20th Street and 20th Avenue. The Blips control the area to the southwest of the Green Line, and the Cruds the area to the northeast.

To prove their virility, the Blips go on "runs" through Crud territory, starting at 1st Avenue and 1st Street and ending at a point on the Green Line that varies from night to night. A run may return to the Green Line in between but never crosses it. A run uses avenues only in the east direction and streets only in the south direction. Thus a run can be described by a string of E's and S's of length 2N-2; such a run ends at Nth Street and Nth Avenue.

The Blips judge the runs made on a given night (all of which have the same length) by how "OG" they are. A run R1 is more OG than a run R2 if and only if:

- R2 returns to the Green Line for the first time at an earlier point than when R1 returns to the Green Line, or
- R1 and R2 return to the Green Line at the same point, but the portion of R1 to that point (ignoring the initial E and final S) is more OG than the portion of R2 to that point (also ignoring the initial E and final S), or
- R1 and R2 return to the Green Line at the same point and are identical to that point, but the rest of R1 is more OG than the rest of R2.

Examples corresponding to these three cases:

- EESS is more OG than ESES.

- EEESSS is more OG than EESESS
- EESSEESS is more OG than EESSESES.

If all the runs of a given length are ordered according to how OG they are, then the rank of a run is its position in the resulting list. EESS has rank 1 and ESES has rank 2.

Your task is to write a program to help the Blips plan and judge their nightly activities. The input to the program is a series of instances followed by 0 0. An instance consists of a line containing a positive integer N, representing the terminus of that night's run (Nth Street and Nth Avenue), followed by positive integer M. The output corresponding to each instance is the run of length 2N-2 of rank M, or ERROR if there are fewer than M runs of length 2N-2.

Sample Input

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

Output for Sample Input

```
EESS
ESES
ERROR
```

Prabhakar Ragde

Problem H: Nice Prefixes

Consider strings formed from characters from an alphabet of size K. For example, if $K = 4$, our alphabet might be $\{a,b,c,d\}$, and an example string is $bbcac$.

For a string S, define $count(S, k)$ to be the number of occurrences of the symbol k in S. For example, $count(bbcac, b) = 2$ and $count(bbcac, a) = 1$.

A prefix of a string S is any string obtained from S by deleting some (possibly none) of the trailing characters of S. For example, the prefixes of acb are the empty string, a, ac, and acb.

A string S has "nice prefixes" if for every prefix P of S and for every two characters $k1$ and $k2$ in the alphabet, $|count(P, k1) - count(P, k2)| <= 2$. For example, $bbcac$ has nice prefixes, but $abbbc$ does not because $count(abbb, b) = 3$ and $count(abbb, c) = 0$.

Count the number of strings of length L on an alphabet of size K that have nice prefixes. This number can be large, so print its remainder when divided by 1000000007.

Input Specification

The first line of input contains a single integer, the number of test cases to follow. Each case is a single line containing the two integers L and K, separated by spaces, with $1 <= L <= 10^{18}$ and $1 <= K <= 50$.

Sample Input

```
1
4 2
```

Output Specification

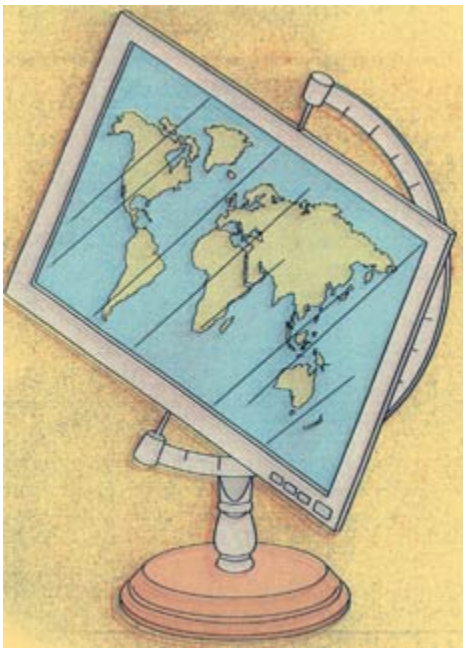
For each case, output a single line containing the number of strings of length L on an alphabet of size K that have nice prefixes, modulo 1000000007.

Output for Sample Input

```
12
```

Malcolm Sharpe, Ondřej Lhoták

Problem I: Point of view in Flatland



Everything is flat in Flatland. The planets are round but they are flat, that is, they are discs in a plane.

The centers of three planets in Flatland are given and their radii. Find the point in Flatland from which all three planets are visible at the same angle, that is, they appear to have the same size measured as angular diameter. Let's call such a point an *isoobservation* point. There can be at most two such points and we are interested in finding the one that gives the largest angular diameter of the planets.

Input consists of several cases, each case is presented at a single line. Each line has nine numbers, three for each disc. Each triple has **x** and **y** coordinates of the disc center and the radius **r** of that disc. The input is terminated by a line with nine zeros and this line should not be processed.

For each case of input, print the **x** and **y** coordinates of the isoobservation point as described above in the format shown in the sample; but if there is no such point, print `No solution`

To simplify the problem you may assume that:

- The discs centers are not all collinear.
- The discs are totally disjoint.
- The discs are transparent and non-refractive. That is, a disc is visible and has the same apparent shape and size, whether or not there's another disc in front of it.
- The input data are such that the existence or non-existence of such a point is computable, even with slight rounding error. But use double-precision, eh?

Sample input

```
10 10 1 30 30 1 50 10 1
0 30 1.0 30 0 1.0 40 40 1.0
10 30 1.0 31 0 1.0 42 43 1.0
10 42 1 62.8 62.8 1 52.5 -25.3 1
10 42 1.1 62.8 62.8 1.2 52.5 25.3 25
0 0 0 0 0 0 0 0 0
```

Output for sample input

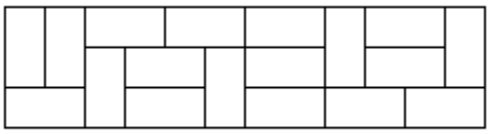
```
30.00 10.00
23.00 23.00
31.58 22.76
49.27 19.73
No solution
```

Piotr Rudnicki

Problem J: Tri Tiling

In how many ways can you tile a 3x*n* rectangle with 2x1 dominoes?

Here is a sample tiling of a 3x12 rectangle.



Input consists of several test cases followed by a line containing *-1*. Each test case is a line containing an integer $0 \leq n \leq 30$. For each test case, output one integer number giving the number of possible tilings.

Sample input

```
2
8
12
-1
```

Output for Sample Input

```
3
153
2131
```

Piotr Rudnicki