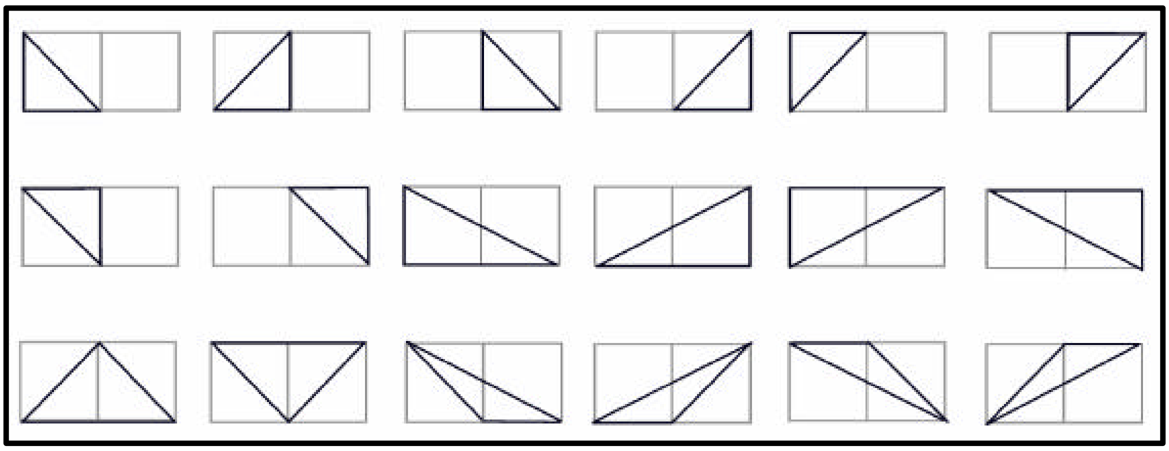
ACM International Collegiate Programming Contest
Dhaka Regional Contest 2005
22nd and 23rd September
Host: North South University
IBM. event sponsor

Problem H

Counting Triangles

Input File: h.in
Output: Standard Output

Triangles are polygons with three sides and strictly positive area. Lattice triangles are the triangles all whose vertexes have integer coordinates. In this problem you have to find the number of lattice triangles in an **M*N** grid. For example in a **(1x2)** grid there are **18** different lattice triangles as shown in the picture below:



Input

The input file contains at most **21** sets of inputs.

Each set of input consists of two integers **M** and **N** **(0<M,N<=1000)**. These two integers denote that you have to count triangles in an **(MxN)** grid.

Input is terminated by a case where the value of **M** and **N** are zero. This case should not be processed.

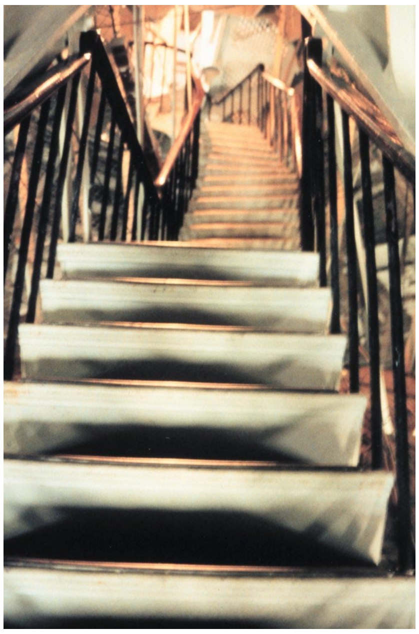
Output

For each set of input produce one line of output. This output contains the serial of output followed by the number lattice triangles in the **(MxN)** grid. You can assume that number of triangles will fit in a **64**-bit signed integer.

Sample Input	Output for Sample Input

```
1 1
1 2
0 0
```
	```
	Case 1: 4
	Case 2: 18
	```

Problem I: Up the Stairs



John is moving to the penthouse of a tall sky-scraper. He packed all his stuff in boxes and drove them to the entrance of the building on the ground floor. Unfortunately the elevator is out of order, so the boxes have to be moved up the stairs.

Luckily John has a lot of friends that want to help carrying his boxes up. They all walk the stairway at the same speed of 1 floor per minute, regardless of whether they carry a box or not. The stairway however is so narrow that two persons can't pass each other on it. Therefore they deciced to do the following: someone with a box in his hands is always moving up and someone empty-handed is always moving down. When two persons meet each other somewhere on the stairway, the lower one (with a box) hands it over to the higher one (without a box). (And then the lower one walks down again and the higher one walks up.) The box exchange is instantaneous. When someone is back on the ground floor, he picks up a box and starts walking up. When someone is at the penthouse, he drops the box and walks down again.

After a while the persons are scattered across the stairway, some of them with boxes in their hands and some without. There are still a number of boxes on the ground floor and John is wondering how much more time it will take before all the boxes are up. Help him to find out!

Input

One line with a positive number: the number of test cases. Then for each test case:

- One line with three numbers N, F, B with $1 \leq N, F \leq 1{,}000$ and $1 \leq B \leq 1{,}000{,}000$: the number of persons, the number of floors (0=ground floor, F=penthouse) and the number of boxes that are still on the ground floor.

- N lines with two numbers f_i and b_i with $0 \leq f_i \leq F$ and $b_i = 0$ or $b_i = 1$: the floors where the persons are initially and whether or not they have a box in their hands (1=box, 0=no box).

Output

One line with the amount of time (in minutes) it will take to get all the remaining boxes to the penthouse.

Sample input	**Sample output**

```
2
3 10 5
0 0
0 0
0 0
2 5 1
2 1
3 0
```
	```
	30
	8
	```

Problem ?

Lift Hopping

Time Limit: 1 second

Ted the bellhop: "*I'm coming up and if there isn't a dead body by the time I get there, I'll make one myself. You!*"

`Robert Rodriguez, "Four Rooms."`

A skyscraper has no more than 100 floors, numbered from 0 to 99. It has **n** (1<=**n**<=5) elevators which travel up and down at (possibly) different speeds. For each **i** in {1, 2,... n}, elevator number **i** takes $\mathbf{T_i}$ (1<=$\mathbf{T_i}$<=100) seconds to travel between any two adjacent floors (going up or down). Elevators do not necessarily stop at every floor. What's worse, not every floor is necessarily accessible by an elevator.

You are on floor 0 and would like to get to floor **k** as quickly as possible. Assume that you do not need to wait to board the first elevator you step into and (for simplicity) the operation of switching an elevator on some floor always takes exactly a minute. Of course, both elevators have to stop at that floor. You are forbiden from using the staircase. No one else is in the elevator with you, so you don't have to stop if you don't want to. Calculate the minimum number of seconds required to get from floor 0 to floor **k** (passing floor **k** while inside an elevator that does not stop there does not count as "getting to floor **k**").

Input

The input will consist of a number of test cases. Each test case will begin with two numbers, **n** and **k**, on a line. The next line will contain the numbers $\mathbf{T_1}$, $\mathbf{T_2}$,... $\mathbf{T_n}$. Finally, the next **n** lines will contain sorted lists of integers - the first line will list the floors visited by elevator number 1, the next one will list the floors visited by elevator number 2, etc.

Output

For each test case, output one number on a line by itself - the minimum number of seconds required to get to floor **k** from floor 0. If it is impossible to do, print "IMPOSSIBLE" instead.

Sample Input

```
2 30
10 5
0 1 3 5 7 9 11 13 15 20 99
4 13 15 19 20 25 30
2 30
10 1
0 5 10 12 14 20 25 30
2 4 6 8 10 12 14 22 25 28 29
3 50
10 50 100
0 10 30 40
0 20 30
0 20 50
1 1
2
0 2 4 6 8 10
```

Sample Output

```
275
285
3920
IMPOSSIBLE
```

Explanation of examples

In the first example, take elevator 1 to floor 13 (130 seconds), wait 60 seconds to switch to elevator 2 and ride it to floor 30 (85 seconds) for a total of 275 seconds.

In the second example, take elevator 1 to floor 10, switch to elevator 2 and ride it until floor 25. There, switch back to elevator 1 and get off at the 30'th floor. The total time is
10*10 + 60 + 15*1 + 60 + 5*10 = 285 seconds.

In example 3, take elevator 1 to floor 30, then elevator 2 to floor 20 and then elevator 3 to floor 50.

In the last example, the one elevator does not stop at floor 1.

Problemsetter: Igor Naverniouk
Alternate solutions: Stefan Pochmann, Frank Pok Man Chu

3292 - Matrissor

Asia - Dhaka - 2005/2006

Matrissor is a special kind of processor which can multiply a sequence of matrices in quick time. It has certain capacity K which means the maximum number of computations (multiplications here) it can perform at one step. For example if K is 1000, then it can multiply 2 matrices of 10 x 10 dimension. But it cannot multiply a (10 x 11) matrix and another (11 x 10) matrix which require 1100 multiplications. There is a limitation of matrissor. It cannot multiply a sequence of matrices optimally. If it is to multiply m matrices, it processes first (m - 1) matrices first and then multiples the resultant matrix with m th matrix.

Your task is to multiply a sequence of matrices optimally using the matrissor with capacity K . Here optimality depends on one criterion. You have to use the matrissor minimum number of times. Say you have 4 matrices available - M_1(10 x 1) , M_2(1 x 10) , M_3(10 x 1) and M_4(1 x 10) . Now if you use a 100 capacity matrissor, then you can multiply M_2 , M_3 and M_4 in one step and in last step you can multiply M_1 , (M_2 , M_3 , M_4). This can be expressed as (M_1 , (M_2 , M_3 , M_4)), where (M_2 , M_3 , M_4) denotes the resultant matrix after multiplying M_2 , M_3 , M_4 .

Input

The input file contains the number of test cases T first, which is at most 30. Each test case begins with a positive integer N ($2 \le N \le 50$) which is the number of matrices. Following N lines contain the dimensions of matrices, one line per matrix. Dimensions will be valid and any dimension will be in between 1 to 50. Next line will contain another integer Q ($1 \le Q \le N$) which is the number of queries, followed by the capacities of the matrissor in one line. Each test case will be followed by a blank line.

Output

For each set of input, print a line `` `Matrix #D' `` in first line, where D is the test case number starting from 1. In next Q lines print the minimum number of steps to multiply all the matrices. If it is not possible to multiply the matrices, then print `` `Impossible.' ``. Put a blank line after each output set. See sample output for details.

Sample Input

```
2
4
10 1
1 10
10 1
1 10
3
100 99 300

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

Sample Output

```
Matrix #1
2
Impossible.
1

Matrix #2
3
2
```

Dhaka 2005-2006

Problemsetter: Md. Kamruzzaman
Special Thanks: Derek Kisman

Problem A: Paper Route

As a poor, tuition-ridden student, you've decided to take up a part time job as a paperboy/papergirl.
You've just been handed your paper route: a set of addresses (conveniently labelled 1 to *N*).

Every morning, you start at the newspaper office (which happens to be address number 0). You have to plan a route to deliver a newspaper to every address - and you also want to get to class right after you're done.
Conveniently, there are only *N* roads in your area connecting the addresses, and each of them takes a known time to traverse.
Also, you've precalculated the time it takes to get to Waterloo campus from each address, including the newspaper office (through some combination of biking, busing, or hitching a ride).
How soon can you be done delivering papers and be in your seat at school?

Input Specification

The input contains multiple test cases. Each test case begins with a single integer *N* (the number of addresses, $1 \le N \le 100{,}000$). This will be followed by *N*+1 lines, each with an integer c_i (starting with $i = 0$, $0 \le c_i \le 1{,}000{,}000{,}000$), the time it takes to get from location *i* to campus. Finally, the next *N* lines will each contain three integers *a*, *b*, *c* ($0 \le a, b \le N$, $a \mathrel{!=} b$, $0 \le c \le 1{,}000$). Each of these lines describes a road between locations *a* and *b* taking *c* minutes to traverse. It is guaranteed that you will be able to reach all the addresses. (Remember that location 0 is the newspaper office.)

Sample Input

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

Output Specification

Output the minimum time it will take to deliver all the papers and get to class.

Output for Sample Input

```
7
```

It's actually better to visit all the addresses, go back to the office, and get to school from there.
An example route: 0 -> 1 -> 0 -> 2 -> 0 -> school = 1 + 1 + 2 + 2 + 1 = 7

Hanson Wang

B

Switch Bulbs

Input: Standard Input
Output: Standard Output

You are given n bulbs and m switches. Each of the switches toggles a list of bulbs. Initially all the bulbs are turned off. Now for a set of desired states of the bulbs calculate the minimum number of switch presses required to reach that state.

Input

Input contains multiple test cases. First line contains an integer T the number of test cases. Each test case starts with a line containing 2 integers n (1≤n≤15) and m (1≤m≤40). Next m line contains the description of m switches. Each of these lines starts with an integer k denoting the number of bulbs that toggles their states after pressing this switch. The rest of the line contains k distinct integers denoting the indices of the bulbs. The bulbs are numbered from 0 to n-1. The next line contains an integer q(1≤q≤5000) that denotes the number of queries. Each of the following q line contains a binary string of length n denoting the desired states of the n bulbs: 1 means the bulb must be on and 0 means the bulb must be off. The rightmost character is the state of bulb 0 and the leftmost character is the state of bulb n-1.

Output

For each test case output contains q+2 lines. First line contains “Case x:” where x is the number of test cases. Each of the next q lines contains one integer denoting the minimum number of switch presses required to reach the bulb states in the i’th query. If the state cannot be reachable by a series of switch presses output -1. The last line will be a blank line.

Sample Input

```
2
3 3
3 0 1 2
2 1 2
1 2
3
101
010
111
4 5
1 0
1 1
2 2 3
3 0 1 3
2 2 3
3
1111
1010
0101
```

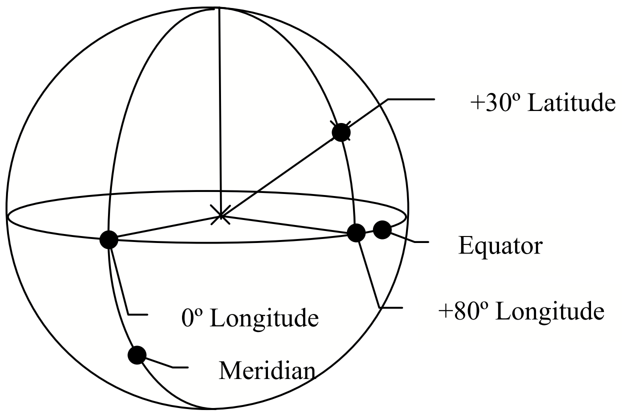
Output for Sample Input

```
Case 1:
3
2
1

Case 2:
3
2
3
```

Problem 5: Communication Planning for Phobos

Life has been found on Phobos, one of the satellites of Mars! Unfortunately, the life forms there aren't quite as advanced as those on Earth, and they don't have modern communications (at least by Earth standards). The Advanced Communication Management Company (ACM) has decided to build a central office and connect the Phobosians' homes for communication (telephone, television, Internet, and so forth). They naturally want to minimize their capital outlay in this effort, and they need to decide how to lay fiber optic cable (essentially on the surface) so the smallest amount is used. Since ACM uses digital broadband technology, it is only necessary that there be a cable path that connects every subscriber and the central office. That is, there does not necessarily need to be a separate cable from the central office to each subscriber's home.



We know the precise location of each Phobosian's home and the planned ACM central office on the surface. These are given using longitude and latitude. Longitude is measured from an arbitrary meridian on the surface of Phobos, and has values in the range −180 degrees to +180 degrees. Latitude is measured from the equator, and has values in the range −90 degrees to +90 degrees. For planning purposes we assume Phobos is perfectly spherical, exactly 16.7 miles in diameter. The figure to the left illustrates one possible location (+80° longitude, +30° latitude).

INPUT

There will be one or more sets of input data. Each set will contain, in order, an integer *N* no larger than 100, but at least 2, followed by *N* pairs of real numbers, each pair giving the unique longitude and latitude, in degrees, of a Phobosian's home or the central office. A single integer zero will follow the last data set.

OUTPUT

For each input data set print a single line containing the data set number (1, 2, …) and the number of miles of cable required to connect all the Phobosian's homes and the central office; show two fractional digits in the distance.

SAMPLE INPUT

```
3
0 0     0 90    0 -90

3
0 0     0 90    90 0

3
0 0     90 0    45 0

6
-10 10   -10 -10   0 0   90 0   80 20   100 -10

0
```

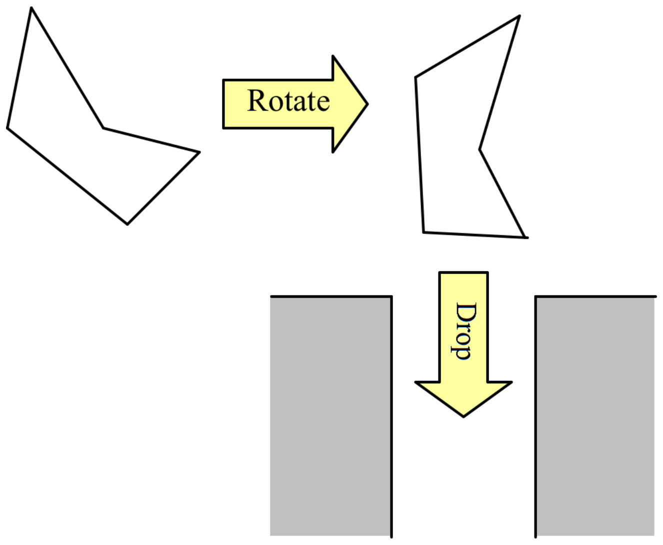
EXPECTED OUTPUT

```
Case 1: 26.23 miles
Case 2: 26.23 miles
Case 3: 13.12 miles
Case 4: 21.16 miles
```

Problem K
Trash Removal
Problem ID: trash

Allied Chute Manufacturers is a company that builds trash chutes. A trash chute is a hollow tube installed in buildings so that trash dropped in at the top will fall down and be collected in the basement. Designing trash chutes is actually highly nontrivial. Depending on what kind of trash people are expected to drop into them, the trash chute needs to have an appropriate size. And since the cost of manufacturing a trash chute is proportional to its size, the company always would like to build a chute that is as small as possible. Choosing the right size can be tough though.

We will consider a 2-dimensional simplification of the chute design problem. A trash chute points straight down and has a constant width. Objects that will be dropped into the trash chute are modeled as polygons. Before an object is dropped into the chute it can be rotated so as to provide an optimal fit. Once dropped, it will travel on a straight path downwards and will not rotate in flight. The following figure shows how an object is first rotated so it fits into the trash chute.



Your task is to compute the smallest chute width that will allow a given polygon to pass through.

Input

The input contains several test cases. Each test case starts with a line containing an integer n ($3 \leq n \leq 100$), the number of points in the polygon that models the trash item.

The next n lines then contain pairs of integers x_i and y_i ($0 \leq x_i, y_i \leq 10^4$), giving the coordinates of the polygon vertices in order. All points in one test case are guaranteed to be mutually distinct and the polygon sides will never intersect. (Technically, there is one inevitable exception of two neighboring sides sharing their common vertex. Of course, this is not considered an intersection.)

The last test case is followed by a line containing a single zero.

Output

For each test case, display its case number followed by the width of the smallest trash chute through which it can be dropped. Display the minimum width with exactly two digits to the right of the decimal point, rounding *up* to the nearest multiple of 1/100. Answers within 1/100 of the correct rounded answer will be accepted.

Follow the format of the sample output.

Sample input

```
3
0 0
3 0
0 4
4
0 10
10 0
20 10
10 20
0
```

Output for the Sample Input

```
Case 1: 2.40
Case 2: 14.15
```

3423 - Bookshelf

North America - North Central - 2005/2006

Agnes C. Mulligan is a fanatical bibliophile. She is continually buying new books. She has a shelf where she puts her newest books. When she decides to read one of these books, she removes it from its location on the shelf, leaving a ``hole" equal to the width of the book. When she buys a new book she places it on the left side of the shelf, which may push one or more books already on the shelf toward the right. This action may cause one or more books to fall off the right end of the shelf.

Your task is to write a program that simulates the addition of books to, and the removal of books from a shelf with a given width. At the end of the simulation, your program is to identify the books on the shelf. Each book has an unique integer identifier in the range 1 to 1000, and has an integral width (in inches) less than or equal to the width of the shelf. A book falls off the shelf if any part of it extends beyond the right end of the shelf.

Input

The input will contain multiple cases, the number of which is specified by the first integer in the input. Each case begins with an integer that specifies the width of the shelf, in inches; this width will be no smaller than 5 inches. This is followed by a sequence of ``events," each on a separate line. The type of event is specified by the capital letter `A' (add a book), `R' (remove a book), or `X' (end the simulation); this letter appears in the first column of the line. For an `A' event the line also contains integers giving the book identifier and its width, in inches. For an `R' event the line also contains the integer book identifier. An `X' event line contains no additional information.

Output

For each case, display the case number (they start with 1 and increase sequentially) and the integer identifiers of the books on the shelf, in order from left to right. Separate the output for consecutive cases by a blank line. Your output should resemble that shown in the sample below.

Sample Input

```
2
10
A 6 5
A 42 3
A 3 5
A 16 2
A 15 1
R 16
X
5
A 1 1
A 2 1
A 3 1
R 2
A 4 1
A 5 1
R 5
R 4
A 6 1
A 7 4
X
```

Sample Output

```
Case 1:
15
3

Case 2:
7
6
```

Clarifications: Integers in this problem will comfortably fit in 32 bits.

North Central 2005-2006