A Pizza Pricing[1]

Pizza has always been a staple on college campuses. After the downturn in the economy, it is more important than ever to get the best deal, namely the lowest cost per square inch. Consider, for example, the following menu for a store selling circular pizzas of varying diameter and price:

Diameter	Price
5 inch	\$2
10 inch	\$6
12 inch	\$8

One could actually compute the costs per square inch, which would be approximately 10.2¢, 7.6¢, and 7.1¢ respectively, so the 12-inch pizza is the best value. However, if the 10-inch had been sold for \$5, it would have been the best value, at approximately 6.4¢ per square inch.

Your task is to analyze a menu and to report the diameter of the pizza that is the best value. Note that no two pizzas on a menu will have the same diameter or the same inherent cost per square inch.

Input

The input contains a series of one or more menus. Each menu starts with the number of options N, $1 \leq N \leq 10$, followed by N lines, each containing two integers respectively designating a pizza's diameter D (in inches) and price P (in dollars), with $1 \leq D \leq 36$ and $1 \leq P \leq 100$. The end of the input will be designated with a line containing the number 0.

Output

For each menu, print a line identifying the menu number and the diameter D of the pizza with the best value, using the format shown below.

Example Input

```
3
5 2
10 6
12 8
2
10 5
12 8
4
1 1
24 33
13 11
6 11
0
```

Example Output

```
Menu 1: 12
Menu 2: 10
Menu 3: 24
```

[1]From the 2011 ACM Mid-Central Regional Programming Contest

B Defending the Fort[2]

You have been just posted to a fort out in the hinterlands. The fort has several cannon emplacements along the enclosing walls. Each emplacement has a store of cannonballs, stacked in a perfect pyramid. Your first job is to find out how many cannonballs each emplacement has.

Cannonballs stack nicely into pyramids. The pyramid can have a square base or a equilateral triangle base. At each level as you go up the size of the square or triangle decreases by 1, until the top level is of size 1. Your job is to calculate the number of cannonballs in each pyramid.

Input

The input consists of $n + 1$ lines of input. Each of the first n lines contains a letter, either "s" or "t" and a positive integer number k representing the number of levels. The letter "s" represents a square pyramid, and the letter "t" represents a triangular pyramid. The last line of input contains the word "end".

Output

Output n lines each containing one number, the number of cannonballs in each pyramid given in the input, in the same order as the input.

Sample Input

```
t 1
s 2
t 3
s 4
t 100
end
```

Sample Output

```
1
5
10
30
171700
```

[2]Modified from the 2008 ACM Pacific NW Regional Programming Contest

C Online Dating

You have been hired by TruLuv Inc. to write part of their online dating system. They have written a state of the art Web 3.0 interface to collect data, but have hit a road block in completing the actual recommendation system.

Your job is to eliminate the obviously unsuitable matches for a given client. Your program is given a list of numbers for each potential match. These numbers represent the client's assessment of the potential match's rating (on a scale from 1 to 10) in categories such as taste in music, dress sense, ability to ride a unicycle, and so on. A candidate is *eliminated* if there is another candidate is at least as good in *every* category, and strictly better in *some* category.

Input

The first line of input contains the number of test cases. The first line of each test case has the number of candidates (at most 100), while the second line has the number of ratings. The ratings for each candidate then follow, one candidate per line.

Output

The output for each test case consist of a line containing the row indices (counting from 1) for the non-eliminated candidates.

Sample Input

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

Sample Output

```
3
1 2 4
```

D The SOCIAL Universe

SOCIAL (Simulation of Consciousness in Artificial Life) is a artificial life simulation in which bugs (Beings whose Universe is a Graph) live in the nodes of a graph. Joe who is lazy wants to make it possible to input a graph in a very simple way, without having to list all edges as pairs of nodes. He came up with the following way to define a graph using a short sequence of positive integers.

The first number, n say, defines the number of nodes of the graph, which are represented by the numbers $1, 2, 3, \ldots, n$. Other numbers in the sequence specify which pairs of the nodes are connected. If d occurs in the sequence and i and j are nodes such that $i + d = j$, then nodes i and j are connected by edge.

It is possible to represent lots of interesting graphs in this way. For example, the sequence $10, 1$ represents a path with ten nodes. The sequence $20, 1, 19$ represents a cycle with 20 nodes. The sequence $15, 1, 5$ represents a 3 by 5 rectangular grid with two extra edges, from node 5 to node 6 and from node 10 to node 11, which makes the graph almost a cylinder.

Michael noticed that not all sequences give a connected universe. There can be pairs of nodes that a bug cannot get from one to the other. For example, the sequence $10, 2, 4$ represents a graph in which the odd nodes are not connected to the even nodes. We do not want the universe to be disconnected like this. Your job is to say when a sequence represents a connected universe, or a disconnected universe.

Input

Each line of input is a sequence of integers that represents a graph as describe above. No graph will have more than one million nodes.

Output

For each line of input there should be one line of output. If the graph represented by the sequence is connected output the word `connected`. If the graph is not connected output the word `disconnected`.

Sample Input

```
10 1
20 2 4
30 3 5 7
1 10
10 11
```

Sample Output

```
connected
disconnected
connected
connected
disconnected
```

E Fly the CoOp

The co-op office is experimenting with a new system for co-op placements. For each student–company pair, they come up with at number representing the total “goodness” of that placement. The goal is to find a valid placement that maximizes the sum of the goodness of all placements. Each candidate should get at most one job, and each position should get at most one candidate.

Input

The first line is the number of test cases. Each test case starts with a line with the number of candidates and the number of jobs. This is followed by a goodness rating for every candidate-job pair in the form “`candidate job goodness`”.

Output

Output for each test case is a single line describing the best placement, with the total goodness followed the “`candidate job`” pairs making up this placement, in sorted order by candidate.

Sample Input

```
2
1 2
1 1 1
1 2 100
2 3
1 1 1
1 2 2
1 3 3
2 1 0
2 2 4
2 3 6
```

Sample Output

```
100 1 2
8 1 2 2 3
```

F NerdCalc3000

You have been hired to write a scientific calculator. In memory of John McCarthy, this calculator will use fully parenthesized prefix expressions. Because big and small numbers are very impressive, and exactitude counts, your calculator must work with arbitrarily large and small rational numbers (integers and fractions), and represent integers and non-integers exactly. Non-integral numbers should be output in "standard form", i.e. x/y where y is a positive integer, and x and y have no common divisors.

Input

Each line of input consists of a fully parenthesized prefix expression. The operations are $+$, $-$, $/$ and $*$. Note that each operator can have an arbitrary number of operands, and that expressions can nest arbitrarily deeply. Non-commutative operations (*i.e.* subtraction and division) should be evaluated starting from the left. You may assume all input expressions are valid, with no division by zero and at least two operands for every operator.

Sample Input

```
(- 1 2 3)
(+ 1 2 3)
(* 1 2 3)
(/ 1 2 3)
(/ (+ 1 2 3) (* 4 5 6))
(* 9999999 9999999 999999 999999 9999999 99999)
(- (+ (- (+ 1 (* 2 3) (- 4 5) (* 6 7)) 2) 4) 11 12)
```

Sample Output

```
-4
6
6
1/6
1/20
9999877000246299833390036419996790001
27
```

G Piles of Blocks

Consider the following puzzle. You are given an m by n grid of squares. You are also given k blocks (that fit exactly on top of one of the squares). At the top of each column and at the left of each row is written a number representing a total for that that row or column. The question is, can you pile blocks on the squares such that the total number of blocks in each row and column adds up to the given totals, while stacking blocks at most 15 high. You are allowed to have blocks left over.

Input

The first line of the input lists the total number of cases. Each case consists of 3 lines. The first line gives the number of rows (at most 10) and columns (at most 10), and the maximum of blocks available (at most 1000), the second line gives the row totals, and the third line gives the column totals.

Output

For each case, if there is no valid block placing, output `NO`, otherwise output a line containing `YES`, followed by an $n \times m$ matrix representing the number of blocks placed in each row and column.

Sample Input

```
3
5 5 15
1 2 3 4 5
1 2 3 4 5
4 4 9
1 2 3 4
1 2 3 4
7 7 58
12 2 12 1 7 9 15
7 7 5 4 12 19 4
```

Sample Output

```
YES
1 0 0 0 0
0 2 0 0 0
0 0 2 0 1
0 0 0 0 4
0 0 1 4 0
NO
YES
7 5 0 0 0 0 0
0 2 0 0 0 0 0
0 0 5 4 3 0 0
0 0 0 0 1 0 0
0 0 0 0 7 0 0
0 0 0 0 1 4 4
0 0 0 0 0 15 0
```