Problem A. Eating Chips

Source file name:	chips.c, chips.cpp, chips.java
Input:	`standard`
Output:	`standard`

Bill and Ted, two brothers, sit down to eat a bowl of N chips while watching the superbowl. Being civilized, they each have their own plate on which they put chips before eating them.

The brothers eat according to a strict set of rules to ensure fairness:

- Both plates start empty.
- Chips shall be consumed from one's own plate, never directly from the bowl, or from the plate of the other brother.
- When Bill fills his plate, he takes B chips from the bowl and places them on his plate. If there are fewer than B chips in the bowl, he takes whatever is left.
- When Ted fills his plate, he takes T chips from the bowl and places them on his plate. If there are fewer than T chips in the bowl, he takes whatever is left.
- During a given timeframe, one of three actions may occur:
  - If both brothers have empty plates, Bill will fill his plate, and then Ted will fill his plate.
  - If one brother has an empty plate, the brother with the empty plate will fill his plate and the other will eat a chip.
  - If neither brother has an empty plate, both brothers will eat a chip.

For a given number of chips in the bowl N, and given constants B and T, output how many chips each brother eats.

Input

The input consists of a series of test cases, one per line. Each line has exactly 3 integers: $1 \leq N \leq 10000$, the number of chips in the bowl, $1 \leq B \leq N$, the number of chips Bill takes at a time, and $1 \leq T \leq N$, the number of chips Ted takes at a time. Input ends on EOF.

Output

For each test case output two integers on their own line: the number of chips that Bill eats, followed by the number of chips that Ted eats.

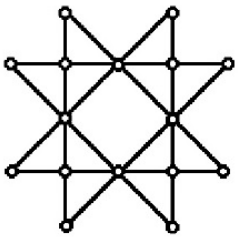
Example

Input	Output
7 1 5	2 5
7 4 5	4 3
7 3 2	3 4

Problem B. Eight Pointed Star

Source file name:	eight.c, eight.cpp, eight.java
Input:	`standard`
Output:	`standard`

You see that star on the right? That's an eight pointed star. It's pretty cool. For example, you can start at any point and draw every segment without ever lifting the pen off the paper or crossing a line that you've already drawn.



See if you can make your computer output one of these stars. Observe the star is made out of 28 separate segments. The horizontal and vertical segments are of length 1, and the diagonal segments have length of $\sqrt{2}$. The lattice points the star intersects are marked by circles. Your goal is to increase the segment lengths of the star by an integer factor, and output the lattice points that the star intersects.

Input

The input consists of a series of at most 25 test cases. On each line is a single positive integer $K \leq 25$, the length of a vertical/horizontal segment of the star (the length of diagonal segments will be $K\ \sqrt{2}$). The input ends on EOF.

Output

For each test case, output the star in all its ASCII glory, with * representing the lattice points. Do **not** output any trailing blank spaces on a line. Do **not** output blank lines between test cases. A star of segment length K will result in $4K + 1$ lines of output. Failing to follow the output format exactly will likely result in your program being judged as incorrect.

Example

Input	Output

3	`   *     *`
	`   **   **`
	`   * * * *`
	`*************`
	` * * * * * *`
	`  ***   ***`
	`   *     *`
	`  ***   ***`
	` * * * * * *`
	`*************`
	`   * * * *`
	`   **   **`
	`   *     *`

Problem C. Eggs

Source file name:	eggs.c, eggs.cpp, eggs.java
Input:	`standard`
Output:	`standard`

John decides one fine day that he's going to throw his entire collection of N eggs at the wall. Now, throwing them one-by-one would be far too boring, so John devises a new system to keep himself entertained. He will throw the eggs in **batches** of possibly differing size. John decides each **batch** size according to the following rules:

1. John dislikes fair splits, so the **batch** size must have fewer than three positive divisors.

2. John dislikes leftovers. Thus, subject to the above condition, the **batch** size must minimize the remainder when the current number of eggs is divided by the **batch** size.

3. John enjoys loud noises. Thus, subject to the above two conditions, the **batch** size must be as large as possible.

Every time John throws a **batch** at the wall, he re-calculates the next **batch** size.

How many **batches** does John end up throwing?

Input

The input consists of a series of no more than 1337 test cases with one integer per line. Each line contains $N \leq 1000000$. Input ends on EOF.

Output

For each test case, output a single integer: the number of **batches** of eggs John throws at the wall.

Example

Input	Output
5 8 10	1 3 2

Problem D. Allergies

Source file name:	allergies.c, allergies.cpp, allergies.java
Input:	`standard`
Output:	`standard`

I have a lot of allergies. I'm allergic to homework, housework, and hard work. I'm allergic to caring, sharing, and swearing. I'm allergic to thinking, drinking, and glasses clinking. And we haven't even begun with my food allergies!

Since I have so many allergies, it'd be helpful to condense them somehow when describing them to someone. Let's consider a universe with a certain number of foods. Each food is described by being free of a list of allergens. If a food is free of *all* my allergens, I can eat it, otherwise I can't.

Here's where my brilliant plan comes into motion. There might be a single allergen that I could claim to be allergic to that would produce the *exact same* set of foods that I could or couldn't eat. For example, suppose I was allergic to (G)luten, (M)ilk, and (E)ggs, and not allergic to (C)asein. The universe has 4 foods: one is ME free (free of Milk and Eggs), another is GE free, the third is GM free, and the last food is GMEC free. Imagine how ingenious I feel when I claim to be allergic to Casein! This single allergen perfectly describes my situation (and entertainingly, is a complete lie).

For a given set of things I'm allergic to, and a universe of foods, can you tell me whether there's a single allergen (which I may or may not be actually allergic to) that perfectly describes my set of allergies?

Input

The input begins with an integer T, the number of test cases. Each test case begins with a non-empty string of unique capital letters - if a capital letter is present in the string, I'm allergic to that ingredient. This is followed by an positive integer $F \leq 1000$, the number of foods. Each of the next F lines contains a single string in capital letters. A food string containing a given capital letter means that it's free of this allergen; no letters are repeated within a string.

Output

For each test case, output a single character if there is a single allergen I can claim to be allergic to that will produce the exact same set of foods I can and can't eat as my real allergies would. If there is no such item, print `No Solution` instead. You are guaranteed that there at most one item that I can claim to be allergic to.

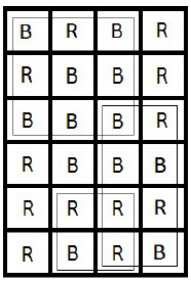
Example

Input	Output
1 GME 4 ME GE GM GMEC	C

Problem E. Boxes of Foxes

Source file name:	boxes.c, boxes.cpp, boxes.java
Input:	`standard`
Output:	`standard`

In a large, rectangular field you spot a horde of foxes. Just tons of them. To your amazement, the foxes are clearly distinguishable into two categories. One type of fox likes math, and such foxes are coloured blue and yellow. The other type of fox likes music, and these are coloured red and white.



You don't like either math or music; instead, you like counting foxes. So here's what you're going to do: first, you construct a region (out of the union of several rectangles) upon which you will count the foxes. Then, perhaps unsurprisingly, you will count how many foxes of each type are in the region.

Input

The input begins with a single integer $T \leq 25$ on its own line, the number of test cases. Each test case begins with three positive integers $M \leq 500$, $N \leq 500$, and $K \leq 13370$: the number of rows and columns in the grid, and the number of rectangles that your region is made out of. Following this are M lines of N characters each: the characters will either be a B , denoting a blue and yellow fox, or a R , denoting a red and white fox. Next come K lines of four integers each, which represent the rectangles your region is made out of. The first two integers represent the $(x,\ y)$ coordinates of the lower left corner, and the last two integers represent the $(x,\ y)$ coordinates of the top right of the box.

The diagram given is a picture of the sample input. Java coders should use **BufferedReader** to read the large input file.

Output

Output one line per test case with two space separated integers: the number of foxes that like math (represented in the input by B), and then the number of foxes that like music (represented by R).

Example

Input	Output
1 6 4 3 BRBR RBBR BBBR RBBB RRRR RBRB 2 1 3 2 3 1 4 4 1 4 3 6	11 7

Problem F. The Yes-Man

Source file name:	yesman.c, yesman.cpp, yesman.java
Input:	`standard`
Output:	`standard`

Nat is a Yes-Man: he just has a hard time saying no to people, almost no matter the circumstance. Due to his easy-going nature, Nat attracts **all** the girls. Unfortunately, that doesn't mean that he wants to date them all - it gets very expensive!

In a given day, Nat will receive requests from N alluring females. Saying yes to a date with girl i doesn't cost Nat energy, but it does cost him C_i dollars. Saying no to a date costs Nat E_i energy (because it's really draining for him to say no), but it doesn't cost him any money.

If Nat uses more than Q total energy or spends more than X total dollars, he will consider himself a "No-Man". This is undesirable. Can you help him find out whether there is a set of dates he can go on to avoid becoming a "No-Man" ?

Input

The input consists of a series of test cases. The input begins with an integer $T \leq 75$, the number of test cases. Each test case takes up three lines. The first line has three positive integers: $N \leq 100$, $Q \leq 1000$, and $X \leq 10000$. The next line consists of N non-negative integers, which represent the energy required E_i to say no to the i^{th} girl. The third line consists of N non-negative integers, which represent the money cost C_i of going out with the i^{th} girl.

Output

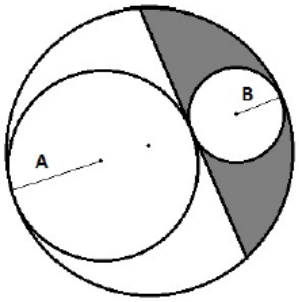
If Nat can get through these date requests without using too much energy or money, output **Yes-Man**. Otherwise, output **No-Man**.

Example

Input	Output
2 1 4 1 3 3 2 4 1 2 3 3 2	Yes-Man No-Man

Problem G. Basic Geometry

Source file name:	basicgeometry.c, basicgeometry.cpp, basicgeometry.java
Input:	`standard`
Output:	`standard`



Competitive programming problems are notorious for having irrelevant preamble. Instead of just getting to the point and asking you to solve a problem, they have long-winded stories about people (who don't matter) doing things (of no particular relevance), just to fill space on the page. It's really quite obnoxious. Why waste everybody's time, when all you want to do is ask a simple problem? Surely problem-writers could do everybody a favour and just skip the paragraphs of irrelevant text that they love to place at the beginning of the problem.

Some experienced problem solvers adopt the tricky strategy of reading the second paragraph first, and then deciding whether the first is worth reading. Alas, this strategy is not foolproof: sometimes even the **second** paragraph of the problem statement is utterly useless! Drat these silly problem-writers. When will they ever learn?

Construct a circle with radius A. Construct another circle of smaller radius B that touches the first. Construct a third circle with radius $A + B$ that touches both of your first two circles and also has its center on the same line as the line between the first two centers. Finally, draw in the tangent to the first two circles inside the largest circle. This will give you the diagram shown above.

Find the area of the shaded region, given A and B.

Input

There will be multiple test cases. Each test case consists of two positive real numbers: $A \leq 50$ and $B \leq 50$, separated by a space, on their own line. Input ends on EOF.

Output

Output to three decimal places (rounded) the area of the shaded region.

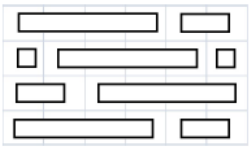
Example

Input	Output
4 1 3 2	8.041 16.771

Problem H. Pole Packing

Source file name:	pole.c, pole.cpp, pole.java
Input:	`standard`
Output:	`standard`

Clayton needs to organize his large collection of poles. Being organized, Clayton would like to pack them all into a single box of size M x N. Each pole has width 1 and a length that is a power of 2 (e.g. 4, 16). Because Clayton likes things to look neat, all the poles must face the same direction. If any pole spans more than one column, there may be no poles that span more than one row, and if any pole spans more than one row, there may be no poles that span more than one column.



One solution to the sample input

Obviously Clayton would prefer to pack these poles efficiently, into as small a box as possible. However, Clayton is a little strange, and insists that the box has a width of M. Please help Clayton out by writing a program that determines the minimum length N required to fit all the poles.

Input

The input consists of a series of test cases. Each test case begins with a line containing positive integers $M \leq 4096$, the fixed dimension of the box. This is followed by a line with a positive integer $K \leq 12$, the number of different pole sizes. K lines follow, each consisting of two positive integers: $1 \leq L \leq 4096$, a pole length, and $Q \leq 65536$, the number of poles of that length. You are guaranteed that all lengths L in the input are unique and powers of 2.

Input ends when $M = 0$. Do not process this as a test case.

Output

For each test case, output the minimum length of the box required to fit all the poles.

Example

Input	Output
4 3 4 4 2 3 1 2 0	6

Problem I. Orbs of Fusing

Source file name:	orbs.c, orbs.cpp, orbs.java
Input:	`standard`
Output:	`standard`

Path of Exile is a interesting take on a Diablo-style action role-playing game (ARPG). The goal of the game (actually, of an ARPG in general) is quite straightforward: kill monsters to acquire better gear so that you can kill stronger monsters (which in turn drop better gear...). It turns out that this is spectacularly addicting.

One thing high quality gear in Path of Exile has is **slots** on an item, which may be connected by **links**. You can think of the S **slots** on an item as being strung in a long line, with $S-1$ possible places to **link** the slots together. In general, the quality of an item is directly related to the maximum chain of connected **slots** on an item (if all the **links** on the item are present, then the maximum chain would have length S, otherwise, it will be less than S).

Here's where the fun begins: one of the cool items dropped in the game is called an "orb of fusing". The orb of fusing re-forges the **links** on an item in the following way: each **link** is randomly generated with probability 0.5, but subject to the constraint that the **links** must change after using the orb. If the re-forging process generates the exact same set of **links** that was already present (this happens with probability $\frac{1}{2^{S-1}}$), then the re-forging process repeats anew.

I currently have F orbs of fusing, and my item has no **links**. I'm going to keep using them on this item until I get a chain of at least L **linked slots** (or until I run out of orbs). Can you tell me the probability that I'll be successful?

Input

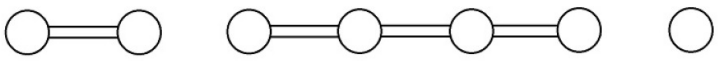
The input begins with an integer $T \leq 30$, the number of test cases. Following this are T lines, each with three positive integers: $2 \leq S \leq 1337$, $F \leq 1337$, and $L \leq S$.

Output

For each test case, output a number rounded to 5 decimal places: the probability that I will attain an item with a chain of at least L **linked slots**.

Example

Input	Output
2 3 1 3 6 10 5	0.33333 0.63862



This is a picture of an item with 7 slots and a maximum chain of 4 **linked slots**

Problem J. Digital Root

Source file name:	digital.c, digital.cpp, digital.java
Input:	`standard`
Output:	`standard`

After a particularly stressful night studying for a class reputed for its killer exams, you fall asleep in a computer lab. Eventually you wake up, only to find yourself locked in. Puzzled, you examine a newly installed digital display by the doorknob. It shows mysterious clues and requires a 10-digit passcode to unlock. An elaborate prank? Could this be your final exam?!

No time for questions, as a high window behind you cracks, letting through a massive wave! Huh, is the university campus submerged underwater? What could possibly have transpired overnight??? This is no accident... whatever mad game your professor is up to this time, it's likely your classmates are trapped inside similar rooms. At this rate, you figure you might drown in about 9 hours. Seek a way out!

Your main clue is the concept of a **digital root**, derived from a non-negative integer by repeatedly replacing the integer by the sum of its digits until the result has only a single digit. For example, the digital root of 1912 is 4 because

$$1912 \to 1+9+1+2 = 13 \to 1+3 = 4.$$

The display presents a string S of digits 0 to 9. Let $|S|$ denote its length. You must insert addition and multiplication signs and parentheses so that the resulting string is an expression E satisfying the recursive grammar (cases separated by '|')

$$E ::= (E+E) \mid (E*E) \mid 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

Thus, you will insert $|S|-1$ pairs of parentheses and $|S|-1$ binary operators + or *, resulting in a total of $|E| = 4|S|-3$ characters. Two ways of doing this are considered to be different if, and only if, the resulting string E is different. Evaluating E as an arithmetical expression yields a well-defined integer result. For $i = 0, ...9$, let n_i count the number of ways to get a result whose digital root is i. The digital root of n_i is the $(i+1)$'th digit of the passcode you need to escape.

Input

Each test case consists of a single line containing the numeric string S ($1 \le |S| \le 40$). The input contains no more than 1000 digits in total.

Output

For each test case, print the corresponding 10-digit passcode on a single line.

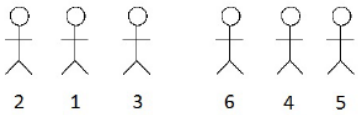
Example

Input	Output
504 20130928	2020110002 2144894475

Problem K. Fanatical Office Mates

Source file name:	fanatical.c, fanatical.cpp, fanatical.java
Input:	`standard`
Output:	`standard`

Dr. Baws has an interesting problem. His N graduate students, while friendly with some select people, are generally not friendly with each other. No graduate student is willing to sit beside a person they aren't friends with.



One solution to the sample input

The desks are up against the wall, in a single line, so it's possible that Dr. Baws will have to leave some desks empty. He does know which students are friends, and fortunately the list is not so long: it turns out that for any subset of K graduate students, there are at most $K - 1$ pairs of friends. Dr. Baws would like you to minimize the total number of desks required. What is this minimum number?

Input

The input begins with an integer $T \leq 50$, the number of test cases. Each test case begins with two integers on their own line: $N \leq 100000$, the number of graduate students (who are indexed by the integers 1 through N), and M, the number of friendships among the students. Following this are M lines, each containing two integers i and j separated by a single space. Two integers i and j represent a mutual friendship between students i and j.

The total size of the input file does not exceed 2 MB.

Output

For each test case output a single number: the minimum number of desks Dr. Baws requires to seat the students.

Example

Input	Output
1 6 5 1 2 1 3 1 4 4 5 4 6	7

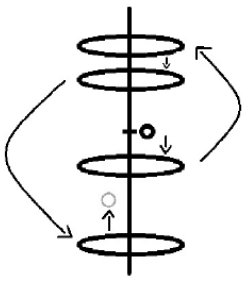
As seen in the diagram, you seat the students in two groups of three with one empty desk in the middle.

Problem L. Portalball

Source file name:	portal.c, portal.cpp, portal.java
Input:	`standard`
Output:	`standard`

Portalball is a futuristic sport in which a ball is dropped from rest (i.e. zero velocity), and onlookers bet on where the ball will come to rest next. In this description, we identify the ball's position with its center of mass, so we can treat it as a point.

Portals come in pairs. If the ball goes through a portal at velocity v, it emerges from its twin at velocity either v or $-v$ depending on whether the two portals have matching or mirroring orientations, respectively. When not going through portals, the ball is accelerated by a controlled constant gravitational field by $980 cm/s^2$ in the negative y direction. In all test cases, it is guaranteed the ball never travels slower than $40 cm/s$ when it's within a millimetre of any portal.



The last sample input.

Input

The input contains up to 100 test cases. The first line of each case contains a single integer N ($1 \leq N \leq 70000$). Each of the next N lines describes a pair of portals by a space-separated list y_1, y_2, o. Integer y_i ($0 < |y_j| < 700000000$) is the position in centimetres of the i'th portal in the pair, relative to where the ball is dropped. All portal positions are distinct. o is either '+' or '−' (quotes for clarity) according to whether the portals have matching or mirroring orientations, respectively. End of input is marked by a 0 on its own line. At most 5 cases will have $N > 1000$.

Output

For each test case, print the ball's position the next time its velocity becomes zero, rounded to the nearest integer, on on a single line following precisely the format of the sample output. If the ball never again comes to rest, print the string `FOREVER FALLING` instead.

Example

Input	Output
1 −5 15 + 1 −5 15 − 2 −5 15 + 10 −20 − 0	FOREVER FALLING STOPS AT 20 cm STOPS AT -10 cm