Problem A
Alice and Bob

ACM Central European Programming Contest, Warsaw 2001, Poland

This is a puzzle for two persons, let's say Alice and Bob. Alice draws an n-vertex convex polygon and numbers its vertices with integers $1, 2, \ldots, n$ in an arbitrary way. Then she draws a number of noncrossing diagonals (the vertices of the polygon are not considered to be crossing points). She informs Bob about the sides and the diagonals of the polygon but not telling him which are which. Each side and diagonal is specified by its ends. Bob has to guess the order of the vertices on the border of the polygon. Help him solve the puzzle.

Example

If $n = 4$ and (1,3), (4,2), (1,2), (4,1), (2,3) are the ends of four sides and one diagonal then the order of the vertices on the border of this polygon is 1, 3, 2, 4 (with the accuracy to shifting and reversing).

Task

Write a program which for each data set:

- reads the description of sides and diagonals given to Bob by Alice,
- computes the order of the vertices on the border of the polygon,
- writes the result.

Input

The first line of the input contains exactly one positive integer d equal to the number of data sets, $1 \leq d \leq 20$. The data sets follow.

Each data set consists of exactly two consecutive lines.

The first of those lines contains exactly two integers n and m separated by a single space, $3 \leq n \leq 10\,000$, $0 \leq m \leq n - 3$. Integer n is the number of vertices of a polygon and integer m is the number of its diagonals, respectively.

The second of those lines contains exactly $2(m+n)$ integers separated by single spaces. Those are ends of all sides and some diagonals of the polygon. Integers a_j, b_j on positions $2j-1$ and $2j$, $1 \leq j \leq m+n$, $1 \leq a_j \leq n$, $1 \leq b_j \leq n$, $a_j \neq b_j$, specify ends of a side or a diagonal. The sides and the diagonals can be given in an arbitrary order. There are no duplicates.

Alice does not cheat, i.e. the puzzle always has a solution.

Output

The output should consist of exactly d lines, one line for each data set.

Line i, $1 \leq i \leq d$, should contain a sequence of n integers separated by single spaces — a permutation of $1, 2, \ldots, n$, i.e. the numbers of subsequent vertices on the border of the polygon from the i-th data set; the sequence should always start from 1 and its second element should be the smaller vertex of the two border neighbours of vertex 1.

Example

For the input:

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

the correct answer is:

```
1 3 2 4
```

Problem B
Binary Stirling numbers

ACM Central European Programming Contest, Warsaw 2001, Poland

The Stirling number of the second kind $S(n, m)$ stands for the number of ways to partition a set of n things into m nonempty subsets. For example, there are seven ways to split a four-element set into two parts:

$$\{1,2,3\} \cup \{4\}, \{1,2,4\} \cup \{3\}, \{1,3,4\} \cup \{2\}, \{2,3,4\} \cup \{1\}$$
$$\{1,2\} \cup \{3,4\}, \{1,3\} \cup \{2,4\}, \{1,4\} \cup \{2,3\}.$$

There is a recurrence which allows to compute $S(n, m)$ for all m and n.

$$S(0,0) = 1; S(n,0) = 0 \text{ for } n > 0; S(0,m) = 0 \text{ for } m > 0;$$

$$S(n,m) = mS(n-1,m) + S(n-1,m-1), \text{ for } n, m > 0.$$

Your task is much "easier". Given integers n and m satisfying $1 \leq m \leq n$, compute the parity of $S(n, m)$, i.e. $S(n, m) \bmod 2$.

Example

$S(4, 2) \bmod 2 = 1$.

Task

Write a program which for each data set:

- reads two positive integers n and m,
- computes $S(n, m) \bmod 2$,
- writes the result.

Input

The first line of the input contains exactly one positive integer d equal to the number of data sets, $1 \leq d \leq 200$. The data sets follow.

Line $i + 1$ contains the i-th data set — exactly two integers n_i and m_i separated by a single space, $1 \leq m_i \leq n_i \leq 10^9$.

Output

The output should consist of exactly d lines, one line for each data set. Line i, $1 \leq i \leq d$, should contain 0 or 1, the value of $S(n_i, m_i) \bmod 2$.

Example

For the input:

```
1
4 2
```

the correct answer is:

```
1
```

Problem C
Calendar of Maya

ACM Central European Programming Contest, Warsaw 2001, Poland

The Classical Maya civilization developed in what is today southern Mexico, Guatemala, Belize and northern Honduras. During its height they developed a sophisticated system for time keeping which they used both to record history and for divinatory rituals. Their calendar consisted of 3 components: the Tzolkin, the Haab and the Long Count.

For divinatory purposes the Maya used the Tzolkin which was composed of 20 day names to which numeric coefficients from 1 to 13 were attached giving a total of 260 distinct combinations. This is the size of the Tzolkin, or ritual, year. From Spanish colonial sources, we know the names of the days:

`Imix, Ik, Akbal, Kan, Chikchan, Kimi, Manik, Lamat, Muluk, Ok, Chuen, Eb, Ben, Ix, Men, Kib, Kaban, Etznab, Kawak, Ajaw`

The sequence of days developed as follows (starting for example at `9 Imix`): `9 Imix, 10 Ik, 11 Akbal, 12 Kan, 13 Chikchan, 1 Kimi, 2 Manik,` ...

The Haab calendar was an astronomical one. It had 365 days divided into 19 months each with 20 days, except the last one which had only 5 days. In a manner similar to the Tzolkin each month name had a number from 1 to 20 indicating the day number within the month. Again, from Spanish colonial sources, we know the names of the months:

`Pohp, Wo, Sip, Zotz, Sek, Xul, Yaxkin, Mol, Chen, Yax, Sak, Keh, Mak, Kankin, Muan, Pax, Kayab, Kumku, Wayeb`

The month Wayeb had just 5 days and was considered an unlucky time of the year.

The Tzolkin and Haab were combined in the inscriptions to create the Calendar Round, combining the 260 day cycle of the Tzolkin and the 365 day cycle of the Haab. A typical Calendar Round date in the inscriptions might be: `3 Lamat 6 Pax`. Note that not all of the combination of days, months and coefficients are possible.

A typical sequence of days in the Calendar Round (starting for example at `3 Lamat 6 Pax`):

`3 Lamat 6 Pax, 4 Muluk 7 Pax, 5 Ok 8 Pax, 6 Chuen 9 Pax, 7 Eb 10 Pax, 8 Ben 11 Pax, 9 Ix 12 Pax, 10 Men 13 Pax, 11 Kib 14 Pax, 12 Kaban 15 Pax, 13 Etznab 16 Pax, 1 Kawak 17 Pax, 2 Ajaw 18 Pax, 3 Imix 19 Pax, 4 Ik 20 Pax, 5 Akbal 1 Kayab, 6 Kan 2 Kayab, ...`

Finally, at the beginning of the Classic Period (AD 200 - 900), the Maya developed an absolute calendar called Long Count which counted the days from a fixed date in the past (the date when the current world was created according to Maya belief). Dates in the Long Count are given (for simplicity) in 5-tuples of the form: `9.2.3.4.5`. Such a date one reads "`9 baktuns 2 katuns 3 tuns 4 winals 5 kins` since the zero date". A "kin" is just one day. A winal is a group of 20 days. A tun is a group of 18 winals (thus a tun has $20 \times 18 = 360$ days, 5 days short of a year). From here on all units come in multiples of 20. Thus a katun is equal to 20 tuns (almost 20 years) and a baktun means 20 katuns (almost 400 years). Thus `9.2.3.4.5` really means "$9 \times 144\,000 + 2 \times 7\,200 + 3 \times 360 + 4 \times 20 + 5$ days since the zero date". Note that for every Long Count date $b.k.t.w.i$ we have $0 \leq k < 20, 0 \leq t < 20, 0 \leq w < 18, 0 \leq i < 20$.

Given the periodicity of the Calendar Round, a legal date such as `3 Lamat 6 Pax` has multiple occurrences in the Long Count. Thus, one difficulty in reading inscriptions is in establishing a date for the inscription when the date is given only in terms of a Calendar Round (very common). In this case one must compute "all" the possible Long Count dates associated with the particular Calendar Round and based in some other context information deduce (for example, the text mentions a king for which other dates are known) which one applies.

We limit our interest to the Long Count dates in the baktuns 8 and 9 (they cover all the Classic Period). We know that the Long Count date `8.0.0.0.0` fell on the Calendar Round `9 Ajaw 3 Sip`.

Task

Write a program which for each data set:

- reads a Calendar Round date,
- computes all Long Count dates in the baktuns 8 and 9 for the given Calendar Round date if this date is legal,
- writes the result.

Input

The first line of the input contains exactly one positive integer d equal to the number of data sets, $1 \leq d \leq 30$. The data sets follow.

Each data set consists of exactly one line that contains exactly one Calendar Round date (maybe illegal): Tzolkin day number, Tzolkin day name, Haab day number and Haab month name separated by single spaces.

Output

For every data set your program must output an ascending sequence of Long Count dates computed for a given Calendar Round date.

The first line of the output for the given input set should contain exactly one integer n equal to the length of the sequence (0, if the input date is illegal).

Each of the next n lines should contain exactly one Long Count date specified by exactly 5 integers (meaning the numbers of baktuns, katuns, tuns, winals and kins respectively) separated by single dots.

Example

For the input:

```
2
3 Lamat 6 Pax
1 Ajaw 9 Chen
```

the correct answer is:

```
15
8.0.17.17.8
8.3.10.12.8
8.6.3.7.8
8.8.16.2.8
8.11.8.15.8
8.14.1.10.8
8.16.14.5.8
8.19.7.0.8
9.1.19.13.8
9.4.12.8.8
9.7.5.3.8
9.9.17.16.8
9.12.10.11.8
9.15.3.6.8
9.17.16.1.8
0
```

Problem D
Decoding Morse sequences

ACM Central European Programming Contest, Warsaw 2001, Poland

Before the digital age, the most common "binary" code for radio communication was the *Morse code*. In Morse code, symbols are encoded as sequences of short and long pulses (called *dots* and *dashes* respectively). The following table reproduces the Morse code for the alphabet, where dots and dashes are represented as ASCII characters "`.`" and "`-`":

A	`.-`	B	`-...`	C	`-.-.`	D	`-..`
E	`.`	F	`..-.`	G	`--.`	H	`....`
I	`..`	J	`.---`	K	`-.-`	L	`.-..`
M	`--`	N	`-.`	O	`---`	P	`.--.`
Q	`--.-`	R	`.-.`	S	`...`	T	`-`
U	`..-`	V	`...-`	W	`.--`	X	`-..-`
Y	`-.--`	Z	`--..`				

Notice that in the absence of pauses between letters there might be multiple interpretations of a Morse sequence. For example, the sequence `-.-..--` could be decoded both as `CAT` or `NXT` (among others). A human Morse operator would use other context information (such as a language dictionary) to decide the appropriate decoding. But even provided with such dictionary one can obtain multiple phrases from a single Morse sequence.

Task

Write a program which for each data set:

- reads a Morse sequence and a list of words (a *dictionary*),
- computes the number of distinct phrases that can be obtained from the given Morse sequence using words from the dictionary,
- writes the result.

Notice that we are interested in *full matches*, i.e. the complete Morse sequence must be matched to words in the dictionary.

Input

The first line of the input contains exactly one positive integer d equal to the number of data sets, $1 \leq d \leq 20$. The data sets follow.

The first line of each data set contains a Morse sequence — a nonempty sequence of at most 10 000 characters "`.`" and "`-`" with no spaces in between.

The second line contains exactly one integer n, $1 \leq n \leq 10\,000$, equal to the number of words in a dictionary. Each of the following n lines contains one dictionary word — a nonempty sequence of at most 20 capital letters from "`A`" to "`Z`". No word occurs in the dictionary more than once.

Output

The output should consist of exactly d lines, one line for each data set. Line i should contain one integer equal to the number of distinct phrases into which the Morse sequence from the i-th data set can be parsed. You may assume that this number is at most $2 \cdot 10^9$ for every single data set.

Example

For the input:

```
1
.---.--.-.-.-.---...-.---.
6
AT
TACK
TICK
ATTACK
DAWN
DUSK
```

the correct answer is:

```
2
```

Problem E
Exchanges

ACM Central European Programming Contest, Warsaw 2001, Poland

Given n integer registers $r_1, r_2, \ldots, r_n$ we define a Compare-Exchange Instruction `CE(a,b)`, where a, b are register indices ($1 \leq a < b \leq n$):

`CE(`a, b`)::`
  `if content(`r_a`) > content(`r_b`) then`
    `exchange the contents of registers` r_a `and` r_b`;`

A Compare-Exchange program (shortly CE-program) is any finite sequence of Compare-Exchange instructions. A CE-program is called a Minimum-Finding program if after its execution the register r_1 always contains the smallest value among all values in the registers. Such program is called reliable if it remains a Minimum-Finding program after removing any single Compare-Exchange instruction.

Given a CE-program P, what is the smallest number of instructions that should be added at the end of program P in order to get a reliable Minimum-Finding program?

Example

Consider the following CE-program for 3 registers:

$CE(1,2)$; $CE(2,3)$; $CE(1,2)$.

In order to make this program a reliable Minimum-Finding program it is sufficient to add only two instructions, $CE(1,3)$ and $CE(1,2)$.

Task

Write a program which for each data set:

- reads the description of a CE-program,
- computes the smallest number of CE-instructions that should be added to make this program a reliable Minimum-Finding program,
- writes the result.

Input

The first line of the input contains exactly one positive integer d equal to the number of data sets, $1 \leq d \leq 10$. The data sets follow.

Each data set consists of exactly two consecutive lines.

The first of those lines contains exactly two integers n and m separated by a single space, $2 \leq n \leq 10\,000$, $0 \leq m \leq 25\,000$. Integer n is the number of registers and integer m is the number of program instructions.

The second of those lines contains exactly $2m$ integers separated by single spaces — the program itself. Integers a_j, b_j on positions $2j-1$ and $2j$, $1 \leq j \leq m$, $1 \leq a_j < b_j \leq n$, are parameters of the j-th instruction in the program.

Output

The output should consist of exactly d lines, one line for each data set.

Line i, $1 \leq i \leq d$, should contain only one integer — the smallest number of instructions that should be added at the end of the i-th input program in order to make this program a reliable Minimum-Finding program.

Example

For the input:

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

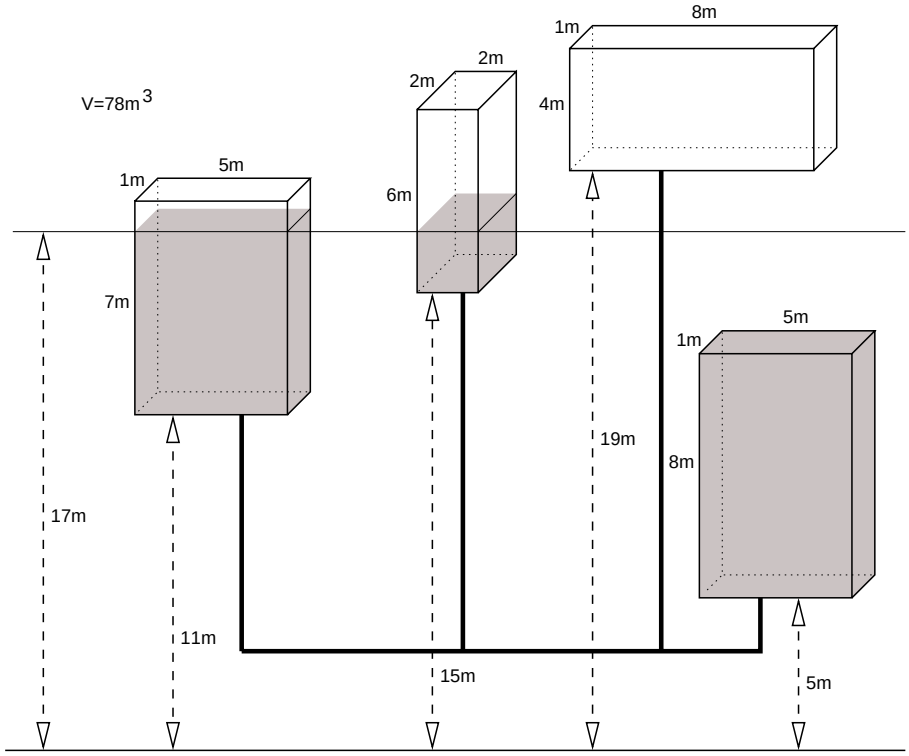
the correct answer is:

```
2
```

Problem F
Fill the cisterns!

ACM Central European Programming Contest, Warsaw 2001, Poland

During the next century certain regions on earth will experience severe water shortages. The old town of Uqbar has already started to prepare itself for the worst. Recently they created a network of pipes connecting the cisterns that distribute water in each neighbourhood, making it easier to fill them at once from a single source of water. But in case of water shortage the cisterns above a certain level will be empty since the water will flow to the cisterns below.



You have been asked to write a program to compute the level to which cisterns will be filled with a certain volume of water, given the dimensions and position of each cistern. To simplify we will neglect the volume of water in the pipes.

Task

Write a program which for each data set:

- reads the description of cisterns and the volume of water,
- computes the level to which the cisterns will be filled with the given amount of water,
- writes the result.

Input

The first line of the input contains the number of data sets k, $1 \leq k \leq 30$. The data sets follow.

The first line of each data set contains one integer n, the number of cisterns, $1 \leq n \leq 50\,000$. Each of the following n lines consists of 4 nonnegative integers, separated by single spaces: b, h, w, d — the base

level of the cistern, its height, width and depth in meters, respectively. The integers satisfy $0 \leq b \leq 10^6$ and $1 \leq h \cdot w \cdot d \leq 40\,000$. The last line of the data set contains an integer V — the volume of water in cubic meters to be injected into the network. Integer V satisfies $1 \leq V \leq 2 \cdot 10^9$.

Output

The output should consist of exactly d lines, one line for each data set.

Line i, $1 \leq i \leq d$, should contain the level that the water will reach, in meters, rounded up to two fractional digits, or the word 'OVERFLOW', if the volume of water exceeds the total capacity of the cisterns.

Example

For the input:

```
3
2
0 1 1 1
2 1 1 1
1
4
11 7 5 1
15 6 2 2
5 8 5 1
19 4 8 1
132
4
11 7 5 1
15 6 2 2
5 8 5 1
19 4 8 1
78
```

the correct answer is:

```
1.00
OVERFLOW
17.00
```

Problem G
Gates

ACM Central European Programming Contest, Warsaw 2001, Poland

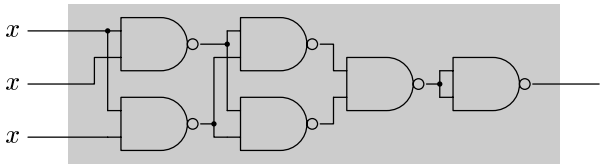
In contemporary VLSI chip industry, the software tools used by electrical engineers perform many optimizations. Your task is to implement one specific optimization of some chip design.

Your tool is given an acyclic net of NAND gates (NAND gate computes the negated conjunction of its inputs, i.e. the output value of the gate is 0 if and only if its both input values are 1). The net is a part of already synthesized component and cannot be changed. All the inputs of the net are connected to one signal x. The objective is to disconnect x from some inputs and to assign constant signals 0 and/or 1 to those inputs in such a way that the function implemented by the design remains unchanged.

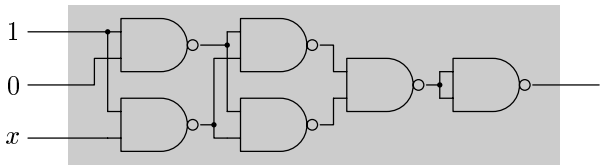
We say that an assignment of x's and/or 0's and/or 1's to the inputs of the net is *optimal* if the number of inputs connected to x is the smallest possible but the net still computes the same function as if all the inputs were connected to x.

Example

Look at the following design.



We can change it to the design with only one variable input, for example:



(Observe that there are other ways of connecting the inputs to just one x and to some number of 0's and 1's that implement the same function).

Task

Write a program which for each data set:

- reads the description of the net,
- computes an optimal assignment of x's and/or 0's and/or 1's to the inputs of the net,
- writes the result.

Input

The first line of the input contains exactly one positive integer d equal to the number of data sets, $1 \leq d \leq 20$. The data sets follow.

Each data set consists of two consecutive lines. The first of those lines contains exactly two positive integers n and m separated by single space, $1 \leq n \leq 100\,000$, $1 \leq m \leq 200\,000$. Integer n is the number of the net inputs and integer m is the number of the gates in the net.

The second of those lines contains exactly $2m$ nonzero integers, separated by single spaces. The numbers on positions $2j-1$ and $2j$ describe the signal sources for the inputs to gate j. The positive number s means the output of gate s. The negative number s means the $(-s)$-th input to the net. The gates and the net inputs are numbered starting from one. The input of each gate is connected to an input of the net or to an output of a gate whose description occurred earlier in the sequence. Each net input is connected to at least one gate input. Each gate output is connected to at least one gate input except the output of the last gate that is connected to the output of the net.

Output

The output should consist of exactly d lines, one line for each data set. The line number i should contain the answer to the i-th data set.

The answer to one data set should consist of a sequence of exactly k characters terminated by the end of line (with no spaces in between). Each of those characters should be `0` (*the digit 'zero'*) or `1` (*the digit 'one'*) or `x` (*lower-case letter 'x'*). The i-th symbol of the sequence denotes the assignment to the i-th input of the net.

If there are more than one optimal assignment then your program should output any of them (but only one).

Example

For the input:

```
1
3 6
-1 -3 -1 -2 1 2 1 2 4 3 5 5
```

one of the correct answers is:

```
10x
```

Problem H
Horizontally visible segments

ACM Central European Programming Contest, Warsaw 2001, Poland

There is a number of disjoint vertical line segments in the plane. We say that two segments are horizontally visible if they can be connected by a horizontal line segment that does not have any common points with other vertical segments. Three different vertical segments are said to form a triangle of segments if each two of them are horizontally visible. How many triangles can be found in a given set of vertical segments?

Task

Write a program which for each data set:

- reads the description of a set of vertical segments,
- computes the number of triangles in this set,
- writes the result.

Input

The first line of the input contains exactly one positive integer d equal to the number of data sets, $1 \leq d \leq 20$. The data sets follow.

The first line of each data set contains exactly one integer n, $1 \leq n \leq 8\,000$, equal to the number of vertical line segments.

Each of the following n lines consists of exactly 3 nonnegative integers separated by single spaces: y_i', y_i'', x_i — y-coordinate of the beginning of a segment, y-coordinate of its end and its x-coordinate, respectively. The coordinates satisfy $0 \leq y_i' < y_i'' \leq 8\,000$, $0 \leq x_i \leq 8\,000$. The segments are disjoint.

Output

The output should consist of exactly d lines, one line for each data set. Line i should contain exactly one integer equal to the number of triangles in the i-th data set.

Example

For the input:

```
1
5
0 4 4
0 3 1
3 4 2
0 2 2
0 2 3
```

the correct answer is:

```
1
```