Problem A. Brackets

Input file:	`brackets.in`
Output file:	`brackets.out`
Time limit:	4 seconds
Memory limit:	256 megabytes

There are k types of brackets in Bytelandian. All these types can be divided into two categories: for some types opening and closing brackets look differently and are represented by different symbols, and for other types opening and closing brackets look the same and are represented by the same symbols.

Any sequence of brackets is called *bracket sequence*. A bracket sequence is *correct* if all its brackets can be divided into pairs and:

- every pair contains opening and closing brackets of the same type;
- the opening bracket appears in the sequence before the closing one of the same pair does;
- a part of the sequence between brackets of the same pair is also a correct bracket sequence.

Note that the sequence of zero elements satisfies this definition.

In other words, a bracket sequence is correct if someone can place numbers and signs of arithmetic operations between adjacent brackets to get a proper arithmetic expression. E. g. the sequence $|()|\lfloor\rfloor$ is a correct bracket sequence because $|5 \times (2+3)| - \lfloor 144/34 \rfloor$ is a notation of number 21.

All in all, till this moment many of you haven't learned anything new about the language of Byteland. But probably you still don't know the most important thing! An *especially correct* bracket sequence is a correct bracket sequence which has no two adjacent identical symbols.

Researchers of Byteland have already made big progress in the study of correct bracket sequences. And now everyone in Byteland knows that the number of correct bracket sequences of length $2n$ for $k = 1$ is either $\frac{(2n)!}{n!\cdot(n+1)!}$ if opening and closing brackets differ, or 1 otherwise.

The question that excites the minds of modern reseachers is much more interesting: they are counting especially correct bracket sequences. Dealing with two categories of bracket types seems to be rather complex for Bytelandian scientists. The research goes step-by-step… Now, for the sake of simplicity, they consider that all types of brackets belong to the first category, i. e. there are k pairs of brackets which are represented by $2k$ distinct symbols. Thus, what is the number of especially correct bracket sequences of length $2n$?

Please help them to find a numerical answer to this question, and don't forget that all cranky Bytelandian researchers were born in the least residue system modulo 1 000 000 007.

Input

The only line contains two integers separated by a single space: the number n of opening brackets in a sequence and the number k of types of brackets ($0 \le n \le 10\,000$, $1 \le k \le 1\,000\,000$).

Output

The only line should contain the remainder of division of the number of especially correct bracket sequences by 1 000 000 007.

Examples

brackets.in	brackets.out
34 1	1
2 3	15
4096 4096	230345484

Problem B. Old Calendar

Input file:	`calendar.in`
Output file:	`calendar.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Kostya has found an old tear-off calendar of the year Y. The calendar has all sheets starting from some date till the 31st of December. There is a date (day, month) and a day of the week written on each sheet. For example, in the 2016 calendar, "Thursday" is written on the sheet corresponding to the 3rd of November.

Kostya wants to determine the year when he can use the remaining sheets of the calendar. The dates and the days of the week must match for all the present sheets. Your task is to determine the minimal such year not less than 2016.

Note that in the range from 2000 to 2099 all the years that are exactly divisible by four are leap years $(2000, 2004, \ldots, 2016, 2020, \ldots)$. Moreover, the jury decided to give you a hint: the answer does not exceed 2099.

Input

The first line contains the number of test cases T $(1 \le T \le 1000)$. Each of the next T lines contains a test case: three integers D, M, Y $(1 \le D \le 31,\ 1 \le M \le 12,\ 2000 \le Y \le 2016)$ which correspond to the first date of the remaining sheets in the calendar of the year Y. Months are numbered from 1 (January) to 12 (December).

It is guaranteed that all the dates are valid.

Output

For each test case, in a separate line print the minimal year Z $(2016 \le Z \le 2099)$.

Example

calendar.in	calendar.out

2
3 11 2016
1 1 2010
	2016
	2021

Problem C. Tickets. Road to the Concert

Input file:	`concert.in`
Output file:	`concert.out`
Time limit:	3 seconds
Memory limit:	256 megabytes

Byteland is a one-dimensional country. All N cities are located along the straight line from the east to the west, the distance between cities i and $i+1$ is equal to one. The capital is located in the city K, and a concert of well-known rock bands is being organized there.

There is a train system in Byteland. A passenger needs to buy a ticket before the trip (of course he can buy another ticket after getting off the train). There are M types of tickets. jth of them can be bought in the city number c_j, it costs a_j, and allows to travel from the city c_j to any city d_j, such that $|c_j - d_j| \leqslant r_j$ (up to r_j stations left or right from the station where the ticket was bought). If the passenger gets off the train, the ticket becomes invalid immediately. *At any time the passenger may have only one valid ticket.*

For the advertising campaign, the concert organizers decided to find out for each city of Byteland the minimum cost of the route from this city to the capital. Help the organizers and find the required values.

Input

The first line of the input contains the number of test cases T ($1 \le T \le 1000$). The next lines describe the test cases. The first line of each test case contains three integers N, K and M ($1 \le K \le N \le M \le 250\,000$). The following M lines of the test case contain three integers c_j, r_j and a_j each ($1 \le c_j, r_j \le N$, $1 \le a_j \le 10^9$).

The sum of values of M over all test cases doesn't exceed $250\,000$.

Output

For each test case, output one line with N integers, ith of which is equal to the minimum cost of the route from the city i to the capital, or -1 if no such route exists.

Example

concert.in	concert.out

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

Problem D. Cubes and Numbers

Input file:	`cubes.in`
Output file:	`cubes.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Lizaveta returned from a trip and gave Pasha an interesting calendar as a gift. In this calendar, the month and the day of the month should be set manually. There are 3 bars with one month written on each of the 4 long faces. Also there are 2 cubes with one digit written on each of the 6 faces. Being an experienced mathematician, Pasha quickly realized that it is not enough to use only two such cubes. But it turned out that he didn't take into account two important nuances: the digit 9 can be obtained by rotation of the digit 6, and there are at most 31 days in each month. Because of these reasons it is enough to use two cubes.

But at once a reasonable question appeared: when other planets are populated, and these planets have another partition of the year into months of some other duration, how many cubes will the locals need for a similar calendar? Also it would be good to create the schemes of these cubes. Let us think that the day of the month should always be shown with the same number of digits (probably with leading zeroes), but this demand doesn't obligate you to use all cubes simultaneously.

Input

The first line contains two integers T and L ($1 \le L \le 18$). Each of T following lines contains an L-digit number N of days in the month ($1 \le N < 10^{18}$) without leading zeroes.

Constraints for L	Constraints for T
$1 \le L \le 6$	$1 \le T \le 1000$
$7 \le L \le 12$	$1 \le T \le 10$
$13 \le L \le 18$	$T = 1$

Output

In the first line output numbers T and L. Futher output T results. In the first line of a result output minimum possible number K of cubes in calendar. In each of K following lines output 6 numbers between 0 and 9, inclusive, placed on the faces of corresponding cube.

Numbers in the same line should be separated by a single space.

Example

cubes.in	cubes.out

1 2
31
	1 2
	2
	0 1 2 3 4 5
	0 1 2 6 7 8

Problem E. Cubes and Numbers 2

Input file:	`cubes2.in`
Output file:	`cubes2.out`
Time limit:	3 seconds
Memory limit:	256 megabytes

Read the statement of the previous problem and write the key part of the checker for that problem. For an output of the previous problem, determine the longest duration of the month that can be served with an offered set of cubes, providing that every day of the month should be shown with exactly L digits (probably with leading zeroes).

Input

The first line contains two integers: the number T of test cases and the number L of digits to show ($1 \le L \le 18$). Then T test cases follow. The first line of a test case contains the number K of cubes ($L \le K \le 30$). Each of the following K lines contains 6 numbers between 0 and 9, inclusive, that are the digits written on the faces of a cube.

Constraints for L	Constraints for T
$1 \le L \le 6$	$1 \le T \le 1000$
$7 \le L \le 12$	$1 \le T \le 10$
$13 \le L \le 18$	$T = 1$

Output

In the first line output the numbers T and L. Then output T results. Each result should be the longest duration of the month that can be served with the offered set of cubes and should be written as an L-digit number (probably with leading zeroes).

Examples

cubes2.in	cubes2.out

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

`1 2`
`2`
`1 2 3 4 5 6`
`1 2 3 1 2 3`
	`1 2`
	`00`

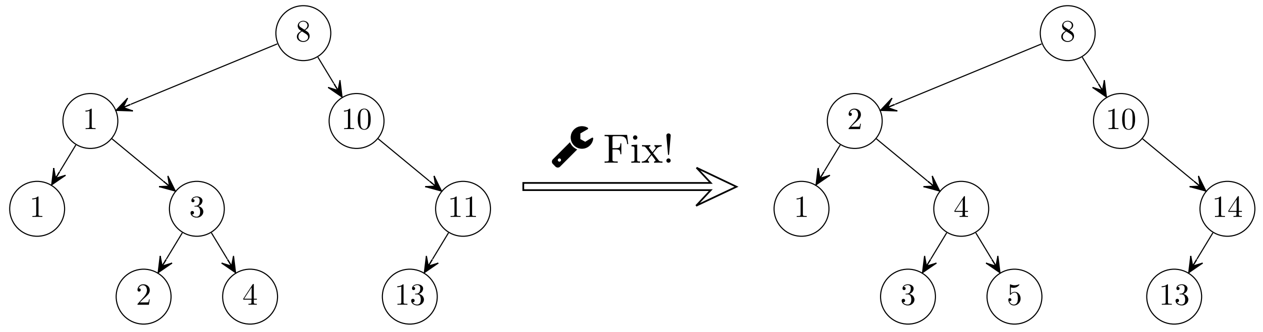
Problem F. Fix the Tree!

Input file:	`fix.in`
Output file:	`fix.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

A *binary search tree* (BST) is an arborescence (i. e. a directed rooted tree with all arcs directed from the root towards leaves), whose vertices each store a key and each have at most two distinguished *children* (adjacent to head ends of arcs), commonly called *left* and *right* and being the roots of the left and the right subtrees correspondingly. The tree additionally satisfies the binary search tree property, which states that the key of each vertex must be greater than all keys stored in the left subtree, and less than all keys in the right subtree.

You develop a database engine and maintain a BST over integer keys in order to perform fast lookups. The tree is stored in a data file. One day the file got damaged due to a hardware failure. You did your best and completely recovered the tree vertices and arcs, but some keys are corrupt (you don't know exactly). The tree is rooted and binary as before, but may have lost its search properties. . .

One obvious way to make the BST valid is to select some vertices and change their keys without rebuilding the tree structure. Note that all the keys should be positive integers. Duplicate keys are not permitted. Compute the minimum number of keys that can be changed to make the BST valid.



Input

Input contains several test cases. For each test case, the first line contains the number N of the vertices in the tree ($1 \le N \le 300\,000$). The ith of the following N lines describes the ith vertex: a key (an integer between 1 and 10^9, inclusive) and an index of the parent vertex (which is 0 for root vertex or a number between 1 and $i-1$, inclusive, for any other vertex, followed by a letter which specifies whether the vertex is the left (`L`) or the right (`R`) child of its parent).

The last line of input contains an integer 0, and this case should not be processed. The sum of N over all test cases in a single file does not exceed 300 000.

Output

For each test case, write a single line containing the minimum number of changes that must be applied to the tree to make it valid.

Examples

fix.in	fix.out
9 8 0 1 1L 1 2L 3 2R 2 4L 4 4R 10 1R 11 7R 13 8L 0	5
3 8 0 5 1R 9 1L 7 4 0 6 1R 2 1L 5 2L 3 3R 1 3L 7 2R 3 1000000000 0 1000000000 1L 999999999 2L 0	2 0 1

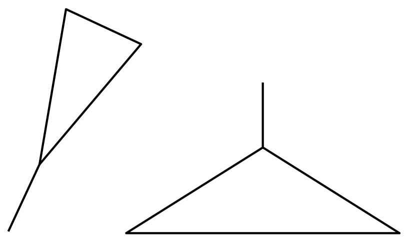
Problem G. The Pear

Input file:	`pear.in`
Output file:	`pear.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

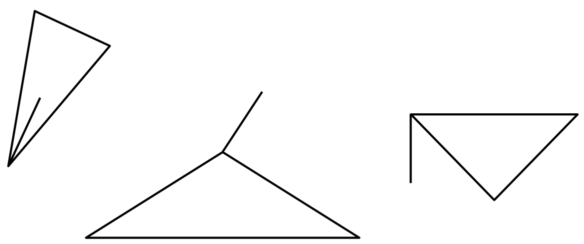
Let's call four segments on the plane *a pear* if three of them form a nondegenerate isosceles triangle, and the fourth segment starts from the apex (the vertex connecting two equal sides) and points towards the direction opposite the altitude dropped from the apex.

The figure *a*) shows examples of pears and the figure *b*) shows examples of the figures that are not considered to be pears.

a)



b)



You are given N distinct segments of nonzero length on the plane. How many various quadruples of these segments form a pear?

Input

The first line contains the number N ($4 \le N \le 300$). Each of the next N lines describes a segment and contains four coordinates of its ends x_1, y_1, x_2, y_2. Coordinates are integers and their absolute values do not exceed 10 000. All the segments are distinct, but they can overlap.

Output

Print the required number of pears.

Examples

pear.in	pear.out

10
0 0 2 0
0 0 1 1
2 0 1 1
1 1 1 2
1 1 1 3
0 0 1 -1
2 0 1 -1
1 -1 1 1
1 -1 1 -2
1 -1 2 -2
	3

4
0 0 2 0
0 0 1 0
1 0 2 0
1 0 1 1
	0

Problem H. Permutations

Input file:	`permutations.in`
Output file:	`permutations.out`
Time limit:	3 seconds
Memory limit:	256 megabytes

Vasya is a student and his research interests include the random shuffling methods. Currently, Vasya explores an algorithm that generates random permutations. Here's how it works.

The array `a[]` of N elements is initially filled with consecutive integers from 1 to N. Then the following procedure performs:

```
for (int i = 0; i < N; ++i) {
    int j = Random(0, N-1);
    Swap(a[i], a[j]);
}
```

The function `Random(x, y)` returns a random integer chosen uniformly from the interval from x to y, inclusive. The function `Swap(x, y)` swaps the values of two variables. The array `a[]` is indexed from zero.

As the result, the array `a[]` contains some random permutation of size N.

Vasya suspects that this algorithm produces permutations in a non-perfectly uniform way, and some of them occur more often than others. Help Vasya, for the given N determine which permutation is the most likely.

Input

The only line contains an integer N ($1 \le N \le 10$).

Output

Print N numbers from 1 to N, the permutation that has the highest probability of occurrence among all permutations. If there are multiple such permutations, print any of them.

Numbers should be separated by single spaces.

Examples

permutations.in	permutations.out
2	2 1
3	2 3 1

Problem I. Big Polygon

Input file:	`polygons.in`
Output file:	`polygons.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Parents gave Basil a big convex polygon as a present for his birthday. Basil was very glad and played with this present for a long time. Once later a pair of scissors caught Basil's eye and he came up with an inspiring thought that playing with many k-gons would be much more amazing than playing with one big n-gon. Therewith it is well-known that k is Basil's favourite number. So he made the decision almost instantly (literally in k nanoseconds). And the scissors together with the big n-gon appeared in his hands. Also Basil decided not to create new vertices, i. e. to cut the n-gon into k-gons with diagonals that don't pairwise intersect in internal points. But, being curious since his childhood, Basil is now interested in the number of ways to accomplish his plan. Meanwhile, Basil thinks that all vertices and of course diagonals are pairwise distinct.

Please, tell Basil the answer to his question, and don't forget that any piece not being a k-gon would upset Basil, so all such cuts shouldn't be counted. Also you shouldn't frighten Basil with a huge number, so don't show him more than 9 last digits.

Input

The only line contains the number n of vertices of the initial polygon and the number k of vertices of every new polygon ($3 \le k \le n \le 4000$).

Output

Output last 9 digits of desired number of ways, or the whole number if it has less than 9 digits.

Examples

polygons.in	polygons.out
8 4	12
25 3	059613650

Problem J. Tickets. Road from the Concert

Input file:	`return.in`
Output file:	`return.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Byteland is a one-dimensional country. All N cities are located along the straight line from the east to the west, the distance between cities i and $i+1$ is equal to one. The capital is located in the city K, and a concert of well-known rock bands is being organized there.

There is a train system in Byteland. A passenger needs to buy a ticket before the trip (of course he can buy another ticket after getting off the train). There are M types of tickets. jth of them can be bought in the city number c_j, it costs a_j, and allows to travel from the city c_j to any city d_j, such that $|c_j - d_j| \leqslant r_j$ (up to r_j stations left or right from the station where the ticket was bought). If the passenger gets off the train, the ticket becomes invalid immediately. *At any time the passenger may have only one valid ticket.*

For the advertising campaign, the concert organizers decided to find out for each city of Byteland the minimum cost of the route from the capital to this city. Help the organizers and find the required values.

Input

The first line of the input contains the number of test cases T ($1 \leq T \leq 1000$). The next lines describe the test cases. The first line of each test case contains three integers N, K and M ($1 \leq K \leq N \leq M \leq 250\,000$). The following M lines of the test case contain three integers c_j, r_j and a_j each ($1 \leq c_j, r_j \leq N$, $1 \leq a_j \leq 10^9$).

The sum of values of M over all test cases doesn't exceed 250 000.

Output

For each test case, output one line with N integers, ith of which is equal to the minimum cost of the route from the capital to the city i, or -1 if no such route exists.

Example

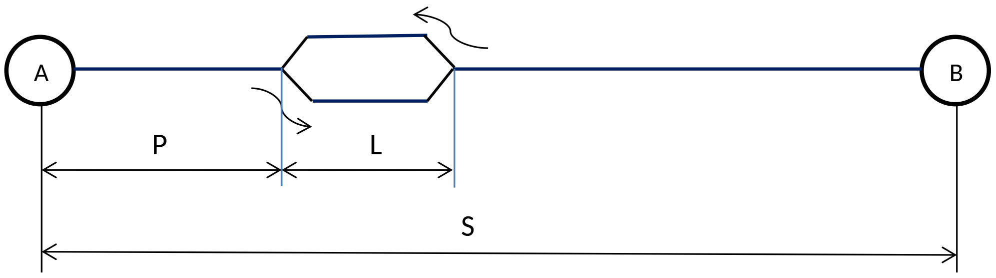
return.in	return.out

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

Problem K. Railway Siding

Input file:	`siding.in`
Output file:	`siding.out`
Time limit:	1 second
Memory limit:	256 megabytes

In Byteland, the railway stations A and B are connected by a single-track railway which has the only double-track passing siding. The length of the track is S km, the siding is L km long, and its closest end is P km away from the station A.



The siding works as follows. A train approaching the siding goes to the right-hand track with no stopping. The exit semaphore is located at the end of each track. The semaphore allows passing only if the track towards the next station has no trains. Otherwise, the train stops at the semaphore and waits for the signal to proceed. For technical reasons the train can pass the siding with the speed of no more than v km/h.

Two trains depart simultaneously from both stations in order to reach the opposite stations. The speed of the train leaving from the station A cannot exceed v_A km/h and the speed of the other train is limited by v_B km/h.

The crew of each train receives a salary for the time spent in trip (including the waiting time). Determine the minimal total time spent by the crews of both trains. The acceleration and deceleration time may be neglected. Also we consider the trains as points.

In Byteland, time and money are rated highly, so calculate the minimal total salary with several digits after the decimal point.

Input

The first line of the input contains the values S, P and L ($P + L < S$). The second line contains the values of v_A, v_B and v.

All the numbers are positive integers and do not exceed 1000.

Output

Output the total time with an absolute or relative error of at most 10^{-9}.

Examples

siding.in	siding.out

`30 10 10`
`70 60 50`
	`1.01904761905`

`30 10 10`
`70 80 50`
	`0.935714285714`

`30 10 10`
`70 60 60`
	`0.952380952381`

`30 25 2`
`70 60 60`
	`1.20714285714`

Problem L. Stickers

Input file:	`stickers.in`
Output file:	`stickers.out`
Time limit:	1 second
Memory limit:	256 megabytes

Vasya has a sticker album that contains N spaces for stickers. Each sticker has its own number and should be stuck in a designated numbered rectangle in the album.

The stickers are sold in blind packs of K distinct stickers, so the purchaser does not know which stickers he is buying. This can mean a collector has to buy huge numbers of packs to complete the collection because of sticker duplication.

Vasya dreams of filling the entire album and tries to predict how many packs of stickers he needs to buy. Help him!

You may assume that each sticker appears with the same chance in each pack (i. e. the company produces the same amount of each and distributes them all at the same time).

Input

Input contains two integes N and K ($1 \le K \le N \le 100$), the total number of stickers and the size of a single pack.

Output

Output the expected number of packs Vasya needs to buy in order to complete his album, with an absolute or relative error of at most 10^{-9}.

Examples

stickers.in	stickers.out
4 2	3.8
100 1	518.737751764