Problem A. Area and Circumference

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

There are N rectangles on a plane, with the edges parallel to the axes. Find among them the one with the highest ratio of area to circumference.

Input

The first line of input contains the integer N ($1 \le N \le 10\,000$) and each of the following N lines describes one rectangle, containing four space-separated integers a, b, c, d ($-100 \le a, b, c, d \le 100$, $a < c$, $b > d$), where (a, b) is the upper left and (c, d) the lower right corner of the rectangle.

Output

The only line of output should contain the maximal ratio of area to circumference among the rectangles, correct to at least 4 decimal digits after the period.

Example

area.in	area.out
3 -13 13 13 -13 9 10 13 9 -11 4 2 -5	6.5

Problem B. Brothers

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

Two integers X and Y are called brothers if they satisfy the following conditions:

- $Y = X + 2010$;
- both X and Y are primes.

Find the number of distinct pairs of brothers in the range from M to N, inclusive. Two pairs are considered distinct if their smaller members are not equal. We also remind that 1 is not considered a prime.

Input

The only line contains the integers M and N ($1 \le M \le N \le 10^9$).

Output

The total number of pairs of brothers.

Example

brothers.in	brothers.out
1 13	0
13 13131313	201166

Problem C. Canonical Binary Tree

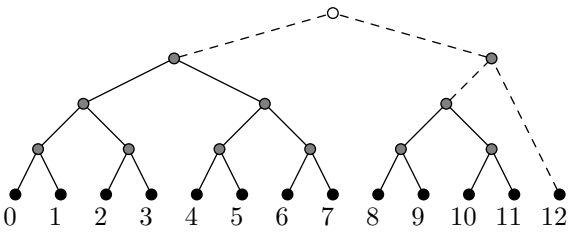
Input file:	`cbt.in`
Output file:	`cbt.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Let's define the following algorithm for building a *canonical binary tree* (CBT) from N leaf nodes (shown in black for $N = 13$ on the figure):

- First, the nodes are collected into complete binary trees, starting from the left and making each tree as big as possible using the nodes still available (adding the gray nodes and solid edges).
- Second, the trees created in the first phase are collected into a single big tree, starting from the right and joining the two smallest trees on each step (adding the white nodes and dashed edges).

It should be easy to see that the structure of a canonical binary tree is fully determined by the number of leaf nodes in the tree.

As with any binary tree, a leaf node in a canonical tree can be designated either by giving its index among the leaves (the numbers on the figure) or by giving the path from the root node to the leaf as a sequence of *left* and *right* commands. For example, the path to the leaf number 4 in the tree shown on the figure would be *left*, *right*, *left*, *left*.



The task is to write a program to translate between these two possible designations of leaf nodes in a canonical binary tree.

Input

The first line of input contains two integers: N ($1 < N < 2^{31}$), the number of leaves in the tree, and Q ($1 \le Q \le 10\,000$), the number of queries to follow.

Each of the Q following lines describes a query. Each query is given by a one-letter query type, followed by a space, followed by a query parameter.

If the query type is `A`, the parameter is an integer I ($0 \le I < N$), the index of a leaf. Leaves are indexed from left to right, starting from zero.

If the query type is `B`, the parameter describes the path from the root to a leaf as a sequence of letters `L` (for *left*) and `R` (for *right*).

Output

The output should contain exactly Q lines, one for each query in the input.

For a type `A` query, the line should contain a sequence of letters `L` and `R`, describing the path from the root to the leaf I.

For a type `B` query, the line should contain a single integer I, the index of the leaf to which the path given as the query parameter leads.

Example

cbt.in	cbt.out
13 2 A 4 B LRLL	LRLL 4

Problem D. Domino

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

A domino bone is a 1×2 rectangle composed of two squares with one of the numbers 0 to 6 marked in each. A full set contains 28 bones, all distinct pairs: $(0, 0)$, $(0, 1)$, ..., $(6, 6)$. A bone with the same number in both squares is called a *double*. Value of a bone is the sum of the numbers in the two squares.

During a game, the bones are laid out in a sequence. The player to move may add a bone from his hand to the sequence if one of the numbers on the bone equals the number in either end of the sequence. For example, with the sequence $(2, 1)$, $(1, 1)$, $(1, 3)$, $(3, 6)$, you could play either a bone with the number 2 on it (adding the bone to the beginning of the sequence) or a bone with the number 6 on it (adding it to the end of the sequence). As a special case, if you have two doubles that match both ends of the sequence, you may play both of them at once. In the previous example, you could play both $(2, 2)$ and $(6, 6)$ in a single move, if you had them. A player who has no valid move, passes.

A group of friends that includes you has been playing with multiple sets of dominoes mixed — thus, a bone can appear in multiple copies. Most of the rules of the game are irrelevant, the one that matters is that after the last move, each player gets a penalty equal to the sum of the values of the bones left in his hand. If the player gets rid of all his bones with the last move, he gets a prize instead...

Now, you are to make your last move. Can you win the prize? If not, at least minimize the penalty.

Input

The first line of input contains two integers: K ($1 \le K \le 1000$), the number of bones in your hand before the last move, and R ($1 \le R \le 1000$), the number of bones already in the sequence.

Each of the following N lines contains two integers from 0 to 6, the description of a bone.

The last line contains $2 \cdot R$ integers, describing the sequence on the table. Description of the sequence from the example above would be `2 1 1 1 1 3 3 6`.

Output

The first and only line of output shoud contain the penalty you get after the last move. If you can win the prize, output -1.

Example

domino.in	domino.out
1 3 1 3 1 3 3 1 1 3	-1
3 4 2 2 4 3 6 6 2 1 1 1 1 3 3 6	7
3 4 2 2 4 3 6 5 2 1 1 1 1 3 3 6	11
3 4 2 2 1 1 6 2 2 1 1 1 1 3 3 1	6

Note: in the second example, the best move is to play the two doubles; in the third example, the bone $(6, 5)$.

Problem E. Express Lines

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

There is a single bus line in a city. The line runs in a closed loop of N stops. All other streets in the city are unfit for bus traffic (and the mayor's office, unfortunately, does nothing to improve them).

The citizens, however, are complaining that the bus rides take too long.

The mayor decided to investigate the possibility to introduce express lines in the city. Each express line goes along the same stops in the same order as the regular line. However, in order to save time, an express line only stops in some subset of the stops and skips the rest of them. Additionally, each express line must follow these two rules:

- it has to stop in at least 2 stops;
- it may stop in at most one of any two stops adjacent on the regular line.

First of all, the mayor wants to know the total number of possible lines satisfying the above conditions. His experts have admitted incompetence and asked you for help. Since they anticipate the answer to be huge, they only want to get it modulo a given integer K.

Input

The first and only line of input contains the integers N ($3 \le N \le 1\,000\,000$) and K ($1 \le K \le 10^{18}$).

Output

The first and only line of output should contain the remainder from division of the total number of possible express lines by K.

Example

express.in	express.out
4 10	2
5 10	5
13 13	0

Problem F. Filter-Art

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

Artist Vasiliy Pupkin used a somewhat unusual method to create his works which he called filter-art. He was working with $M \times N$ cm photographic plates and several $k \times k$ cm (where k is a fixed positive integer) semi-transparent filters. The first of the filters decreased the luminosity by 1 unit, the second by 2 units, etc.

The artist laid the filters on a photographic plate so that the edges of the filters were parallel to the edges of the plate and stood at some integral number of centimeters from them. Each filter was fully on the plate (not dangling over the edge), but the filters may have partially (never completely, though) overlapped with one another. The luminosity on the areas covered by several filters was decreased by the amount equal to the sum of the decrease values of the overlapping filters. After the filters had been laid, the artist exposed the plate and as the result obtained a grayscale image with several shades of gray.

Unfortunately, after the death of the artist, all his filters were lost...

An art historian studying one of the works of Pupkin would like to determine the number of filters the artist used to create it as well as the layout of the filters. As he is not up to the task himself, he asks for your help.

Input

The first line of input contains the integers M and N ($1 \le M, N \le 500$). A matrix with M lines and N columns follows, giving the result of digitizing the photo-art image. Each value in the matrix shows the decrease of luminosity of light on the corresponding area of the plate, compared to the full beam (so, the value zero means there were no filters applied to that area). The values are non-negative integers and do not exceed 10^9.

Output

The first line of output should contain an integer L, the number of filters used. Each of the following L lines should give the position of the top left corner of one filter relative to the top left corner of the plate (row position first, column position second, as in the example). The filters should be listed in the increasing order of their luminosity decrease values. If there are several solutions, output any one of them.

Example

filter.in	filter.out
4 5 1 4 4 3 0 1 6 6 5 0 1 6 6 5 0 0 2 2 2 0	3 0 0 1 1 0 1

Note: the filters had the size 3×3.

Problem G. Game

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

There is a new TV game show looking for participants...

The game is as follows:

There are M balls in the lottotron. Each of the balls is marked with a unique integer from 1 to M and has two buttons that look the same. After a button is pressed and released, it returns to the initial state and after that it is impossible to tell which of the two buttons was pressed. However, one of the buttons triggers a blue and the other a red light in the studio.

All the balls are equally likely to be dropped from the lottotron.

A player waits for a ball to be dropped and presses one of the buttons on the ball. Neither of the lights is triggered at once, but the player's choice is recorded by the lottotron. The ball is then returned to the lottotron and the procedure is perfomed once more. After a commercial break (couldn't do without, could we ⌣̈) the lights corresponding to both buttons pressed by the player are lit. If the two lights are of the same color, the player wins; if not, he loses...

The two balls have been dropped, and the two buttons have been pushed... Now, while the commercials are running, here's a puzzle for you to solve:

You have magically learned that the number on one of the balls that dropped from does not exceed K and on that ball the player pushed the button corresponding to the blue light. What is the probability that the player wins?

Input

The first line of input contains a single integer N ($1 \le N \le 1000$), the number of test cases. Each of the following N lines describes one test case and contains two integers K and M ($1 \le K \le M \le 10^9$).

Output

For each test case output on a separate line one real number, the probability of the player winning, accurate at least to 10^{-9}.

Example

game.in	game.out
2 1 313131313 131313 131313	0.5 0.3333333333

Problem H. Headache

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

A department head of a hospital invited two professors, **A** and **B**, for medical consultations. There are N patients in the department, all of whom have a headache and have to see both professors.

The patients are of three different types:

1. those who have to see professor **A** first and **B** after that;
2. those who have to see professor **B** first and **A** after that;
3. those who can see the two professors in either order.

The department head can estimate the duration of visit of each patient to each of the invited professors based on the medical history of the patient.

As the professors charge by time, the department head wants to minimize the total billable time. The professors bill for the time spent seeing the patients, but also for the time spent waiting for the next patient. The professors arrive together and also leave together.

(The small print in the contract states that the professor who finishes first will also be paid for the time spent waiting for his colleague to finish.)

The task is to write a program to order the patients so as to minimize the total time billed by the professors.

Input

The first line of input contains the integer N ($1 \le N \le 20$). Each of the following N lines describes one patient as three integers: the type of patient (from 1 to 3), the duration of visit to professor **A**, and the duration of visit to professor **B**. The duration of each visit is a positive integer and does not exceed 1000.

Output

The first line of output should contain the total time billed by the professors. The second line should contain a permutation of the integers $1 \ldots N$ — the order in which professor **A** will see the patients. The third line should contain the order for professor **B**.

If there are several optimal solutions, output any one of them.

Example

headache.in	headache.out
3 3 5 7 1 6 1 2 2 6	28 1 2 3 3 1 2
1 3 13 13	52 1 1

Problem I. "Injurious" Triples

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

There is a sequence $(a_1, \ldots, a_N)$ of N positive integers. It is known that all elements of the sequence are distinct. Let's call (a_i, a_j, a_k), where $1 \le i < j < k \le N$, an *"injurious" triple* if $a_i + a_k = 2 \cdot a_j$.

Check if the given sequence contains at least one "injurious" triple. If so, find it.

Input

The first line of input contains the integer N ($1 \le N \le 10\,000$). The second line contains N positive integers not exceeding 1 313 131, the elements of the sequence.

Output

The first line of output should contain either "`Yes`" or "`No`" (without the quotes), depending on whether the sequence contains "injurious" triples. If it does, the second line should contain the space-separated integers i, j, k, giving the location of one of the "injurious" triples in the sequence.

Example

injurious.in	injurious.out
5 10 20 30 40 50	Yes 2 3 4
13 1 3 13 31 131 313 1313 3131 13131 31313 131313 313131 1313131	No

Note: input file for the second example doesn't contain line break between numbers 13131 and 31313.

Problem J. Jellous balls

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

Two jellous balls, with radii R_1 and R_2, are moving on an infinite horizontal plane with constant deceleration[1]. The magnitudes of deceleration are a_1 and a_2, respectively. The coordinates of centers and velocities of the balls at the moment $t = 0$ are (x_1, y_1, R_1), (x_2, y_2, R_2) and $(vx_1, vy_1, 0)$, $(vx_2, vy_2, 0)$, respectively. If balls touch each other, they become inseparable. Compute the time when the balls will touch each other.

Input

Input contains up to 100 test cases. The first line of input contains T, the number of test cases. Then test cases follow, each given on two consecutive lines. The first line contains R_1, a_1, x_1, y_1, vx_1, vy_1, the parameters of the first ball. The second line contains the same data for the second ball. All the values are integers, with $1 \le R_i \le 1000$, $0 \le a_i \le 1000$, $-1000 \le x_i, y_i, vx_i, vy_i \le 1000$. In the test cases where the balls will not touch, the distance between them never becomes less than $2 \cdot 10^{-7}$.

Output

Output should contain one real number on a separate line for each test case, the time until the balls will touch, correct to at least 4 decimal digits after the period. If the balls already touch at the time $t = 0$, output 0. If the balls never touch, output -1.

Example

jellous.in	jellous.out
3 1 0 4 0 -1 1 1 0 0 4 1 -1 5 3 0 0 7 4 5 2 6 -8 2 5 13 13 13 13 13 13 1 3 1 3 1 3	1.292893218813452 0 -1

[1] Once a ball's speed reaches zero, the ball stops and no longer moves.

Problem K. Kth P-Partition

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

Let's define the *P-partition* of the integer N as a sequence of positive integers $(p_1, \ldots, p_M)$ that satisfies the conditions:

- $\sum_{i=1}^{M} p_i = N$,
- $|p_i - p_j| \le 1$ for all i and j.

Two P-partitions that differ only by the order of elements are considered distinct.

Each P-partition may be represented as a string where the elements of the partition are separated from each other by single spaces. The P-partitions may be sorted into the lexicographic order of their string representations[2].

For example, consider the lexicographic order of all P-partitions of the integer 5:

```
1 1 1 1 1
1 1 1 2
1 1 2 1
1 2 1 1
1 2 2
2 1 1 1
2 1 2
2 2 1
2 3
3 2
5
```

Find the lexicographically K-th P-partition of the given integer N.

Input

The first line of input contains two integers N ($1 \le N \le 60$) and T ($1 \le T \le 1000$). T is the number of test cases and all the test cases are about the one value of N. Each of the following T lines describes one test case and contains one value of K ($1 \le K \le 10^{18}$).

Output

For each test case output on a separate line the lexicographically K-th P-partition of the integer N. If the total number of P-partitions of N is less than K, output -1.

Example

kth.in	kth.out
13 3 1 13 1313	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 -1

[2]The space character is considered less than any digit.

Problem L. Lacking Spaces

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

Basile, a sixth-grader, is learning the basics of programming, tutored by his brother Peter, a college freshman. He has been given the task to print in a single line all the integers from 1 to N, for a given N.

Unfortunately, Basile forgot to print the spaces to separate the numbers. So, for $N = 13$, the result was `12345678910111213`.

Peter was disappointed to see that and said: "Since you now have one big number instead of N small ones, why don't you compute the remainder from dividing it to another given integer M."

Soon Peter realized that he had spoken too fast, as not only Basile, but also Peter himself was unable to solve the problem... Now he asks for your help!

Input

The only line of input contains the numbers N and M, separated by a space. The limits are as follows: $1 \le N \le 10^{18}$, $1 \le M \le 10^9$.

Output

The required remainder.

Example

lacking.in	lacking.out
13 13	4
12345678910 1000000000	345678910

Problem M. Munich

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

There are several types of tickets used for public transport in Munich:

- day tickets for adults that are valid from the moment they are marked until midnight and allow unlimited number of rides for one person; the price of such a ticket is p_1;
- day tickets for children that work like the adult ones, except only children aged from 5 to 14 years can use them (of course, a child can also ride with an adult ticket); the price of such a ticket is p_2;
- day tickets for groups that allow up to five adults to ride on a single ticket, or any of the adults to be replaced by one or two children; thus, also two adults and six children, or a group of ten children could travel on a single group ticket; the price of a group ticket is p_3;
- similar tickets valid for three days, priced q_1, q_2, and q_3, respectively.

There are also single-ride tickets but those are too expensive to even consider...

A tourist group of M adults and N children has just arrived in Munich for K ($1 \le K \le 3$) days and plans to make heavy use of public transport. They will always travel together. What is the minimal total amount they must spend on tickets?

Input

The first line of input contains M, N, K; the second line — p_1, p_2, p_3; the third — q_1, q_2, q_3. All the values are integers from the range $1 \ldots 1000$.

Output

The amount to be spent on tickets.

Example

munich.in	munich.out
13 1 3 13 1 31 31 13 131	279