Problem A. Angle Patterns

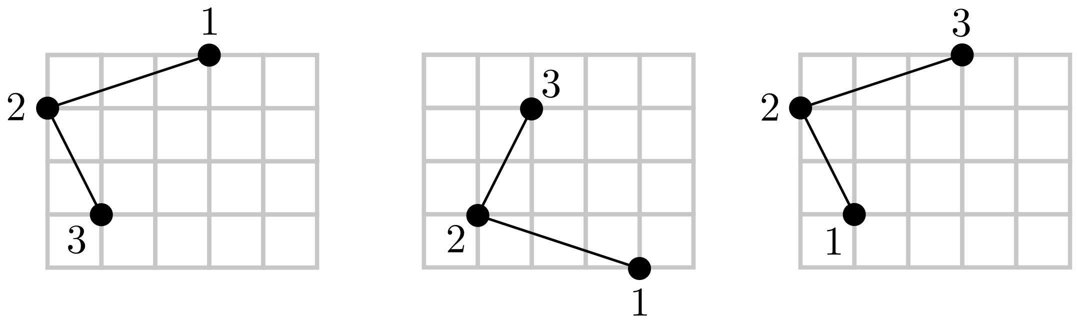
Input file:	`angle-patterns.in`
Output file:	`angle-patterns.out`
Time limit:	1 second
Memory limit:	256 mebibytes

Little Vasya took a sheet of paper and checked it with N vertical and M horizontal line segments, obtaining a grid of square cells. Then he used a marker to draw an angle pattern: he chose three different points of segment intersection and connected the first point with the second one and the second point with the third one.

Vasya's father is good at mathematics, so he immediately came up with an extra challenge for his son: how many different angle patterns can you draw on this sheet of paper? Degenerate patterns, which have all points on the same line, must also be considered.

Two patterns are considered the same if one can be obtained from the other by applying zero or more elementary transformations each of which is either a translation or a reflection against a vertical or horizontal line (note that rotation is not considered an elementary transformation). The points with the same numbers must coincide after the transformations.

Look at the pictures for clarity. Here we have a grid formed by six vertical and five horizontal lines, thus N is 6 and M is 5. The first and the second angle patterns are equal. The third pattern differs from the two others (notice the order of points).



Initially Vasya tried to count the number of patterns manually, but then stopped for some reason. Help him to get the answer to his dad's question.

Input

The only line of input contains two integers N and M $(1 \le N, M \le 1000)$.

Output

Output the number of distinct angle patterns that can de drawn on the grid.

Examples

angle-patterns.in	angle-patterns.out
1 3	3
3 2	24

Note

There are three points in the grid for the first sample. Let us label them consecutively as A, B and C. Three different patterns which could be drawn here are $A - B - C$, $B - A - C$ and $C - A - B$. Please note that, despite the "shape" of the $B - A - C$ and $C - A - B$ patterns is similar, they have different order of points. Patterns $C - B - A$, $B - C - A$ and $A - C - B$ are the reflections of corresponding former patterns.

Problem B. Best Strategy

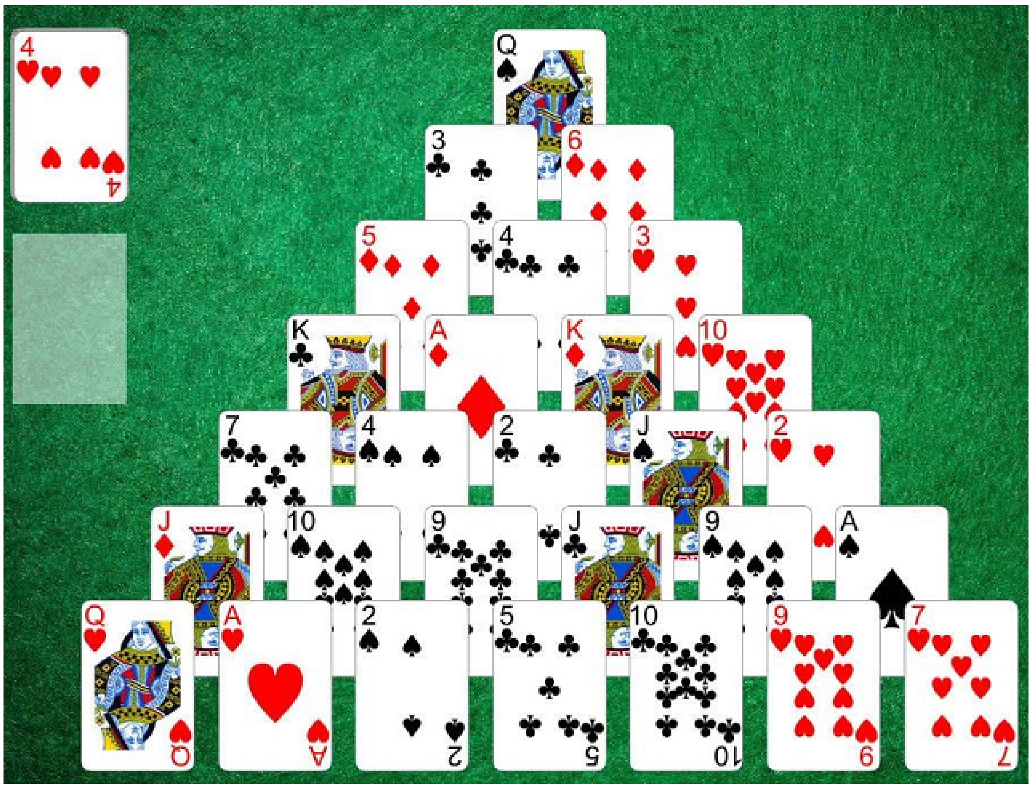
Input file:	`best.in`
Output file:	`best.out`
Time limit:	1 second
Memory limit:	256 mebibytes

Sometimes Yandex employees take part in card game tournaments. In this problem, we describe the rules of one such game.

Pyramid is a solitaire game where the object is to get all the cards from the pyramid to the foundation. The object of the game is to remove pairs of cards that have the sum of their values equal to the most valuable card in the deck. Initially, a pyramid of 28 cards is arranged on the table.

When using the common French deck, Aces value at 1, Jacks value at 11, Queens 12, and Kings 13. So the highest value is 13.

To set up the pyramid, deal one card face up at the top of the playing area, then two cards beneath and partially covering it, then three beneath them, and so on until you have dealt out a row of seven cards, for a total of 28 cards dealt. The picture below shows an example of the pyramid.



The remaining cards are placed in the *Stock* at the side of the table.

To play, pairs of exposed cards can be removed to the Foundation if the sum of their values is 13. Additionally, Kings can be removed immediately to the Foundation. The cards which are being removed must not be covered by the other cards or by each other. For example, when an Ace rests on a Queen, these two cards can not be removed as a pair.

You may draw cards from the Stock one at a time in the order they are placed there. After a card is drawn, you can match it with any exposed card if the sum of their values is 13. If no match is possible, the drawn Stock card is still discarded to the Foundation.

Once the Stock is exhausted and no more pairs can be made, the game ends.

To calculate the score for the game, count the number of remaining cards in the pyramid. The perfect score is therefore zero, achieved if all cards from the pyramid have been removed to the Foundation.

You are given the order of cards in the pyramid and the Stock. Find the best achievable score (as small as possible).

Input

The first line of input contains 28 cards in the order they are placed to the playing area (from top to bottom row, each row from left to right). The second line contains 24 cards in the order they are placed in the Stock (from the top to the bottom).

You may assume that cards are marked as `A`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `T`, `J`, `Q`, `K`, and their values range from 1 to 13 in that order. A card of each type appears exactly four times in total.

Output

Output a single line with the best achievable score.

Example

best.in	best.out

`T2Q787K35TK469423QKQ87J496A4`
`5Q392JT6ATKJ67A83585JA92`
	`6`

Note

Rules of the game are based on an article from Wikipedia, the free encyclopedia.

Problem C. Coin Game

Input file:	`coins.in`
Output file:	`coins.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

After lunch, Vasya's and Petya's pockets commonly become full of coins received as change. Therefore they enjoy the following game.

They put coins in a row in some order and then make moves in turn. As a move, one can:

- take a few coins in a row starting with the leftmost;
- take a few coins in a row starting with the rightmost;
- choose an integer d ($d > 1$), leave each d-th coin (that is, coins at positions d, $2 \cdot d$, $3 \cdot d$, …, counting from the leftmost coin) and take the rest.

The game ends when there are no coins left. The goal of each player is to maximize the amount of money taken.

Games played according to these rules used to pass too quickly, so the players invented an additional restriction: in a move, you cannot take more than some appointed amount of money in total.

Vasya and Petya laid out all the coins. Help them to determine how many each of them gets if both players play optimally, and Vasya starts the game.

Note that Vasya and Petya use modern Russian coins. We would like to remind you that in Russia, coins in denominations of 1, 5, 10, 50, 100 (1 ruble), 200 (2 rubles), 500 (5 rubles) and 1000 kopeks (10 rubles) are in circulation.

Input

Input contains several test cases. The number T ($1 \le T \le 20$) of test cases is given in the first line of input. Then the test cases follow.

Each test case is described by two lines. The first one contains two integers N and M ($1 \le N \le 250$, $1000 \le M \le 10\,000$): the number of coins put in the row and the limit of money per move in kopeks. The second line contains N integers a_1, a_2, …, a_N where each is one of $\{1, 5, 10, 50, 100, 200, 500, 1000\}$.

The sum of values of N over all test cases is not greater than 1000.

Output

For each test case, output a single line with two numbers: the amounts of kopeks Vasya and Petya win if they both play optimally.

Example

coins.in	coins.out

3
5 1000
1 1 1 1 1
8 1000
1 1000 1 1000 10 1000 5 1000
5 1000
1 1000 5 1000 10
	5 0
	3000 1017
	1016 1000

Problem D. Domination

Input file:	`domination.in`
Output file:	`domination.out`
Time limit:	5 seconds
Memory limit:	256 mebibytes

Consider a sequence a_i of N integers.

Two types of operations are allowed:

- Set the value x for all sequence elements in the range from index L to index R inclusive.
- Find the dominating element of the part of the sequence that begins at index L and ends at index R inclusive.

The element b is called *dominating* in the sequence if more than a half of the sequence elements are equal to b.

Given the initial sequence, perform the operations and print the results.

Input

The first line of input contains the number N ($1 \le N \le 200\,000$) of elements in the sequence. On the second line, there are N values a_i ($0 \le a_i < 10^6$). The third line contains the number Q ($1 \le Q \le 200\,000$) of operations. Each of the next Q lines contains a description of a single operation. Operations of the first type are given as "`set` L R x" where $1 \le L \le R \le N$ and $0 \le x < 10^6$. The second type operations are described as "`query` L R" where $1 \le L \le R \le N$. All numbers are integers.

Output

For each operation of the second type, output a line containing a dominating value if it exists, or -1 if it does not.

Example

domination.in	domination.out

`10`
`1 2 1 2 1 2 1 2 1 2`
`10`
`query 1 10`
`query 2 10`
`query 1 9`
`set 1 5 3`
`query 2 3`
`query 1 10`
`query 1 9`
`set 1 10 1`
`query 2 3`
`query 1 10`
	`-1`
	`2`
	`1`
	`3`
	`-1`
	`3`
	`1`
	`1`

Problem E. Equation

Input file:	`equation.in`
Output file:	`equation.out`
Time limit:	4 seconds
Memory limit:	256 mebibytes

You need to find all positive integer solutions of the equation $n^x \equiv x \pmod{m}$ for the given integers n and m.

Input

The only line contains two integers n and m ($1 \le n \le m \le 1\,000\,000$).

Output

The only line must contain a set of all positive integer solutions of the equation in the form of union of the minimum possible number of disjoint subsets $\{a_i k + b_i \mid k \ge 0\}$ ($a_i > 0$, $b_i > 0$) each of which is an infinite arithmetic progression. Each subset must be written like "`{` a_i`k + `b_i `}`". Use the character `U` (the uppercase Latin letter `u`) as the union operation sign. All subsets must be printed in the order of non-decreasing of a_i, and subsets with equal a_i must be printed in the order of increasing of b_i. If there are more than 50 000 subsets, output only the first 50 000 of them and the text "` U ... (and `S` more)`" where S is the corresponding integer denoting the number of subsets that were not printed.

It is guaranteed that for any correct input, the result can be represented in the form described above.

Examples

equation.in
`2 7`
equation.out
`{ 21k + 11 } U { 21k + 15 } U { 21k + 16 }`

equation.in
`3 7`
equation.out
`{ 42k + 2 } U ... (and 5 more)`

Note

The output for the second test case must contain all six subsets, but was shortened for the purpose of demonstration of the case of more than 50 000 subsets.

Problem F. Safe Flight

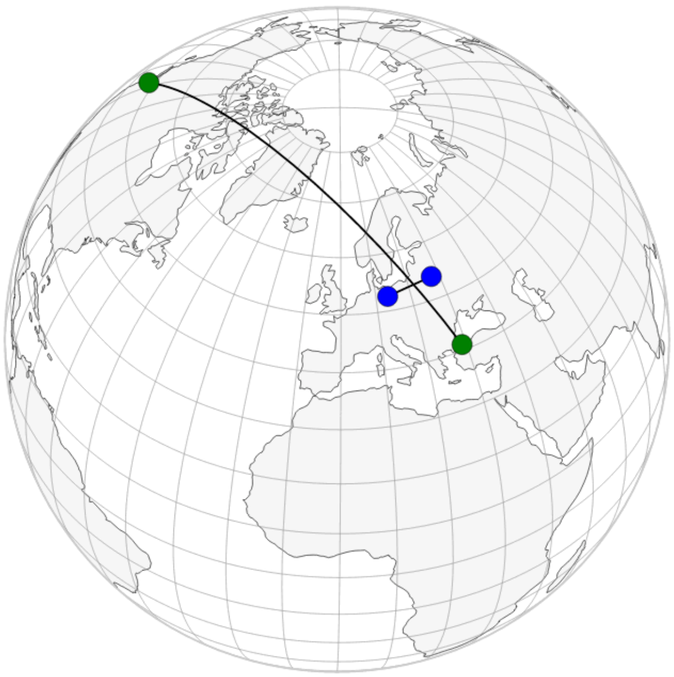
Input file:	`flight.in`
Output file:	`flight.out`
Time limit:	1 second
Memory limit:	256 mebibytes

Employees of the international company Yandex often have to fly on business trips between different offices of the company. And, of course, it is impossible to think only about work in the sky, so they discuss everything that they see around.

Ivan and Simon wondered how safe it is actually to travel by plane, because a huge number of flights are being performed at the same time. Of course, air traffic controllers maintain the safe, orderly and expeditious flow of air traffic and prevent aircraft collisions. Nevertheless, an interesting math problem was revealed.

You are given the coordinates of the cities of departure and arrival for two flights. Determine whether the routes of the aircrafts cross or not.

In this problem, we assume that the Earth is a unit sphere, and the planes fly by the shortest route.



Input

The first line of input contains the number T ($1 \le T \le 1000$) of test cases. Each of the next T lines describes one test case. Each test case sets positions of start and finish points for the first and the second flight.

A point is specified using two consecutive numbers: latitude and longitude (angles measured in degrees). Latitude is the geographic coordinate that specifies the north-south position of a point on the Earth's surface. The equator has a latitude of $0°$, the North Pole has a latitude of $+90°$, and the South Pole has a latitude of $-90°$. Longitude specifies the east-west position of a point on the Earth's surface. The Prime Meridian, which passes through Greenwich, establishes the position of $0°$ longitude. The longitude of other places is measured as the angle east or west from the Prime Meridian, ranging to $-180°$ westward (inclusive) and to $+180°$ eastward (exclusive).

Hence, each line specifying a test case contains eight real numbers $\text{lat}_{1,s}$, $\text{lon}_{1,s}$, $\text{lat}_{1,f}$, $\text{lon}_{1,f}$, $\text{lat}_{2,s}$, $\text{lon}_{2,s}$, $\text{lat}_{2,f}$ and $\text{lon}_{2,f}$ separated by single spaces, where $-90 \le \text{lat}_{i,s}, \text{lat}_{i,f} \le 90$ and $-180 \le \text{lon}_{i,s}, \text{lon}_{i,f} < 180$

for $i \in \{1, 2\}$ which denote the flight number; s stands for start and f stands for finish. All the numbers are given with no more than three digits after the decimal point. Exponential notation is not allowed.

It is guaranteed that the departure and arrival cities are not located at diametrically opposite points and do not coincide.

All test cases satisfy the following condition: the distance on the sphere between flight trajectory and departure and arrival city of other flight is at least 10^{-7}.

Output

For each test case, print a single line. Write `DANGER` if the routes cross, or `SAFELY` otherwise.

Example

flight.in

2
53.906 27.555 52.516 13.377 41.009 28.967 47.679 -122.132
21.286 -156.602 21.092 121.245 -17.136 49.591 -33.542 116.474

flight.out
DANGER SAFELY

Note

Consider the first example. The first aircraft operates a short-haul flight from Minsk (Belarus) to Berlin (Germany), and the second wide-body airliner makes a transatlantic flight from Istanbul (Turkey) to Redmond (USA). The flight paths cross in the sky over Poland.

The second test case describes two airliners, one flying over the Pacific Ocean, the other flying over the Indian Ocean. So they are never close.

Problem G. Version Control System

Input file:	`git-hashes.in`
Output file:	`git-hashes.out`
Time limit:	1 second
Memory limit:	256 mebibytes

In a certain Yandex LLC unit, there are N developers. Each of them has one string pattern which he considers beautiful (a pattern is made of uppercase letters of Latin alphabet). An employee can post his favorite pattern on his personal page if he thinks it is necessary. The patterns of all developers have the same length L.

Nowadays, one of the most popular version control system is Git. A particular source code version (a *commit* in Git terms) is identified by a hash string composed of digits from `0` to `9` and letters from `a` to `f`. We say that a hash matches the given pattern if it contains the same characters on the places where the pattern has the same characters. For example, the hash `1ac1ca1` matches the patterns `ACEAOXA` and `ABCACBP`.

Some day, Peter, the development unit head, wondered how many Git hashes there are that match all the patterns of unit employees. Soon he found the answer. Then he became interested in how many hashes match the patterns of employees working on particular projects. There are Q project teams, each is a subset of the unit's developers. Using a list of members of each team, find the number of hashes that match all the team's patterns at the same time. Print the number for each team in the form $X \cdot 2^Y$ where X is an odd integer (that is, 48 should be printed as $3 \cdot 2^4$).

Input

There are two integers on the first line of input: N and L ($1 \le N, L \le 200$). Next N lines describe N patterns so that line $i+1$ contains i-th employee's favorite pattern. Each pattern has length L and consists of uppercase Latin letters.

The next line contains an integer Q ($1 \le Q \le 200$). Each of the next Q lines contains a string of length N made up with `0`s and `1`s. If the i-th developer ($1 \le i \le N$) works in the team described by the current string, then the i-th character is `1`, otherwise, it is `0`. It is guaranteed that there is at least one `1` on each line.

Output

For each of the Q project teams, output the number of hashes that match the favorite patterns of all the members of the team. Use the form specified above and print X and Y separated by the characters "`*2^`".

Example

git-hashes.in	git-hashes.out
3 7 ABCDCBA XXXTPPP OAOAOAO 2 101 011	1*2^8 1*2^4

Problem H. Hack

Input file:	`hack.in`
Output file:	`hack.out`
Time limit:	1 second
Memory limit:	256 mebibytes

The newbie hacker Vasya dreams of becoming an employee of a major international IT company. For the present, while surfing the net, he came across a certain KiloBank. The title suggested to him that the bank is not so new, and its security system may be not up-to-date. That's exactly what Vasya was looking for! After some time, the hacker got access to the database containing usernames and password hashes. There is only one little thing left: bruteforce the original passwords and perform fraudulent transactions.

First of all, of course, we need to figure out how the hash function works. It turns out that the password can be up to several megabytes long! The database stores the password length unencrypted in the first field for every user. The second field looks like a simple 32-bit hash of the password. But the developers apparently understood that this is not enough, and even a large bit width of the hash function sooner or later will not be able to keep secrets from hackers like Vasya. So the developers invented the third field. It is unknown how Vasya could guess the meaning of that field, but he managed to do it! He realized that π-function is computed for a password, then its value is hashed and written to the database as the third field.

Given the password P and the position i from 1 to the length of P, the value $\pi(i)$ is defined as follows: it is the length of the longest substring of P starting strictly after the first character and ending at i-th character that is equal to a prefix of P of the appropriate length. In particular, $\pi(1)$ is always zero since only the empty substring satisfies the definition.

"Probably it is even possible to compute the function fast...", Vasya thought, but couldn't find too much time for this problem. Now, to move on, Vasya wants to understand how many different π-functions exist for strings of length N (considering that there is no limit on the size of the alphabet: we are allowed to make up strings from an infinite set of characters).

After digging in publicly available sources, he realized that the problem had already been studied. A sufficiently efficient algorithm for finding that number had not been discovered yet.

However, Vasya thought for a while and decided that there are only few strings for which $\pi(i)$ exceeds K at any position i, so they can be completely ignored. Thus, he now wants to count the number of different π-functions such that all their values are at most K. Would it simplify the problem?

Vasya was born and grew up in the field $\mathbb{Z}_{1\,000\,000\,007}$, therefore, he is interested only in the remainder of division of the number he seeks by $10^9 + 7$.

Input

The only line of input contains two integers N and K ($1 \le N \le 10\,000\,000$, $0 \le K \le 10$): the length of the string and Vasya's upper bound.

Output

Output the number Vasya is interested in: the number of different π-functions for strings of length N such that all their values are at most K, taken modulo $10^9 + 7$.

Example

hack.in	hack.out
5 2	17

Problem I. Optimal Choice

Input file:	`independent.in`
Output file:	`independent.out`
Time limit:	1 second
Memory limit:	256 mebibytes

You are given a connected undirected graph. Each vertex has a certain weight W_i and, moreover, belongs to no more than one simple cycle. You are to find the maximum-weight independent set: a set of vertices in the graph such that no two vertices in the set are adjacent, and their total weight is the maximum possible.

Input

The first line of input contains the number of vertices N ($1 \le N \le 100\,000$) and the number of edges M ($0 \le M \le 200\,000$) of the graph. The second line contains N integers W_i ($1 \le W_i \le 1000$) which are the weights of the vertices. The next M lines describe the edges as pairs of integers u_i, v_i where $1 \le u_i, v_i \le N$ and $u_i \ne v_i$. There is at most one edge between any pair of vertices.

It is guaranteed that the graph is connected and each vertex belongs to no more than one simple cycle.

Output

Output the maximum weight of the independent set.

Example

`independent.in`	`independent.out`

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

Problem J. Jeopardy

Input file:	`jeopardy.in`
Output file:	`jeopardy.out`
Time limit:	1 second
Memory limit:	256 mebibytes

Young boy Innokentiy is fond of games about dungeons. In one such game, his character is in jeopardy inside an endless empty dungeon. Every point of the dungeon might be either lit or dark. Initially, all points of the dungeon are under the veil of darkness.

There is a Cartesian coordinate system in the dungeon such that Innokentiy's character is located at the point with coordinates $(0, 0)$. We may assume that Innokentiy's character has infinitely small foot size and thus occupies an infinitely small area. In order to escape from the dungeon, Innokentiy needs to lead his character through the lit area of the dungeon to some exit.

The character of Innokentiy is able to cast N magic spells. The i-th spell can illuminate an arbitrary rectangle with sides of lengths A_i and B_i such that they are parallel to the coordinate axes. It does not matter which of the sides has length A_i and which B_i. After a spell has been cast, all points that are inside the rectangle or on its border become lit until the end of the game. Unfortunately, each spell can be used no more than once. Rectangles for different spells may overlap freely.

Innokentiy knows the coordinates of Q possible exits from the dungeon. Help him to discover which exits are reachable and which are not. For the exits that can be reached, print the minimum number of spells which have to be used.

Input

The first line of input contains the number of spells N $(1 \le N \le 20)$. Each of the next N lines contain a pair of integers A_i and B_i $(1 \le A_i, B_i \le 10^7)$ separated by a space: the sides of a rectangle i-th spell can light.

The next line contains the number of exits Q $(1 \le Q \le 10)$. Each of the following Q lines contains a pair of integers X_j, Y_j $(1 \le X_j, Y_j \le 2 \cdot 10^9)$: the j-th exit's coordinates.

Output

For each of the Q exits, print a line containing the minimum number of spells which is enough to get out from that exit, or -1 if the exit cannot be used.

Example

jeopardy.in	jeopardy.out

3
1 4
2 2
5 1
4
9 2
10 2
2 10
10 5
	2
	3
	3
	-1