

Problem D. Wild Card: Subway

Input file: `stdin`
Output file: `stdout`
Time limit: 2 s
Memory limit: 256 Mb

Bob trains to play Dota 2 for a long time, but his team has not been able to get to the Starladder 2014 finals. Bob was terribly upset and was thinking about resignation from cybersport, when his teammate told him the good news: the organization has provided the Wild Card (extra place for the tournament) to the Bob's team. But there was another problem — there not much time before the finals, Bob needs to get to the train station as soon as possible and arrive at the venue. The fastest way to get there — to use the local subway. Bob has ran into the subway and began to look for an opportunity to purchase a ticket. Fortunately, there were recently installed ticket machines in the subway stations. Machine operates as follows. Firstly machine takes a certain amount of coins. If this amount is not enough, the machine doesn't do anything (yes, it is impossible to get coins back). If there are enough coins to buy one ticket, the machine calculates the amount of money that needs to return to Bob. If machine can return that amount (using coins in a machine storage), it returns proper amount of money and a ticket. Otherwise, it goes into an error state, and gives nothing (even a ticket).

But let's go back to Bob. He was going to throw a few coins into the machine, but worried that he may not receive a ticket. Knowing about the set of coins in the machine and the set of coins that Bob has, determine, is there are any chance, that after throwing into the machine some of his coins (maybe all coins) it will be enough to buy a ticket, but the machine couldn't give change.

Ticket machine can also use just inserted coins. It is guaranteed that Bob has enough money to buy a ticket.

Input

The first line contains three positive integers: N — number of Bob's coins, K — the number of coins in the machine at the time of Bob arrival, and M — subway fare ($1 \leq N, K, M \leq 5000$).

The second line contains N integers — nominal of Bob's coins.

The third line contains K positive integers — nominal of the coins that are in the machine.

They are ordered by ascending of the nominal. Coins nominal does not exceed 10000.

Output

If Bob can get a ticket by throwing any amount (that sufficient to purchase) to the machine, print the word **Go**.
If Bob can get an error state while working with the machine, print the word **Wait**.

Example

stdin	stdout
3 2 4 1 2 3 1 2	Go
3 2 4 1 2 3 2 4	Wait

Problem E. Wild Card: Bus

Input file: `stdin`
Output file: `stdout`
Time limit: 2 s
Memory limit: 256 Mb

Having failed to buy a ticket on the subway (see problem «Wild Card: Subway»), Bob decided to take a bus to get to a railway station. City where Bob lives is a set of stops and bidirectional roads connecting them. The bus can move along the roads between stops. Bob knows the amount of time it takes to cross each road. Also, the bus can spend any non-negative amount of time on the bus stop.

Bob has played Dota2 for years, it made him famous in the city. One of the Bob's fans, Ivan, says he has got Bob's autograph on one of the bus stations. He doesn't remember exactly when it happened, but he says it happened at some time between t_b and t_e . Not everyone believed his words, though. To refute Ivan's words, some even studied the videos of surveillance cameras installed at some stations. Knowing the time required to get from one station to another and the cameras information determine whether Bob could give Ivan autograph on the specified station in supposed time interval.

Input

First line contains two numbers N, M — number of bus stops in the city and the number of roads, respectively ($1 \leq N \leq 1000, 0 \leq M \leq \frac{N(N-1)}{2}$).

Next M lines describe the map of the city. Every line has numbers a_i, b_i, l_i ($1 \leq a_i < b_i \leq N, 1 \leq l_i \leq 10\,000$) — numbers of stops joined by the road and the time required for bus to cross this road. Each pair is connected by not more than one road. Every road is bidirectional.

Next line has three integers C, t_b and t_e ($1 \leq C \leq N, 0 \leq t_b \leq t_e \leq 15\,000\,000$) — station number, specified by Ivan, the earliest and the latest time (inclusively) at which Bob could leave an autograph.

Next line contains one integer K ($1 \leq K \leq 50\,000$) — the amount of data cameras.

Next K lines have pairs of integers p_j, t_j ($1 \leq p_j \leq N, 0 \leq t_j \leq 15\,000\,000$) — station number and time the camera detected Bob riding in a bus. All time stamps t_j are different. The bus driver is in no hurry, unlike Bob, so his way isn't optimal, moreover, the bus can drive through the same road several times during one ride.

Output

If Bob could give Ivan his autograph in specified time interval on the specified bus station, print **YES**, otherwise print — **NO**.

Example

stdin	stdout
4 6 1 2 10 2 3 10 3 4 10 1 4 10 1 3 15 2 4 15 2 12 25 2 1 10 3 30	YES
4 6 1 2 10 2 3 10 3 4 10 1 4 10 1 3 15 2 4 15 2 12 15 2 1 10 3 25	NO

Problem F. Windrunner at Your Service

Input file: `stdin`
Output file: `stdout`
Time limit: 2 s
Memory limit: 256 Mb

Bob trains to play Dota2. Now he plays as a hero called Windrunner. One of the abilities of the hero — Shackleshot — allows you to bind an enemy to a tree. To perform binding requires the following:

- distance from the hero to the enemy should not exceed $D1$
- distance from the enemy to the tree should not exceed $D2$
- the angle between the «hero-enemy» and «enemy-tree» should not exceed A degrees in absolute value.

Bob knows position of his hero (X_w, Y_w) , the position of the enemy (X_e, Y_e) , as well as the position of all N of the trees on the map (X_i, Y_i) . We assume that the enemy is not moving. Help Bob to determine what is the minimum distance need to pass by his hero in order to bind the enemy to one of the trees.

Input

The first line contains three integers — $D1, D2$ and A ($1 \leq D1, D2 \leq 1000, 1 \leq A \leq 90$). The second line contains four integers — X_w, Y_w, X_e, Y_e . The third line contains only one integer number N ($1 \leq N \leq 1000$) — the number of trees located on the map. The following N lines are the coordinates of trees — X_i, Y_i . All coordinates (hero, enemy and all trees) are integers and do not exceed 1000 in absolute value.

Output

If Bob can bind the enemy to a tree, it is necessary to print the minimum path which hero need to pass, with an absolute or relative error of not more than 10^{-6} . Otherwise, print -1 .

Example

stdin	stdout
1 2 30 0 0 1 1 1 2 2	0.4142135624
1 2 30 0 0 1 1 1 3 3	-1
2 2 30 0 0 1 1 1 2 2	0.0000000000

Problem G. What is the Answer?

Input file: `stdin`
Output file: `stdout`
Time limit: 2 s
Memory limit: 256 Mb

Bob learned to play Dota2 very well. Now Bob has decided to participate in the Challenge 24 — popular programming competition. On qualification round he met a task, for which scores are awarded with depending on how well does participant's solution is effective relatively to others solutions. Effectiveness of the solution is evaluated by a natural number A — absolute score of solution. The smaller number shows the better solution. Score for the task (relative score) is calculated with the following formula:

$$R = [100 * (1 - \sqrt{(1 - BA/A)})]$$

where BA — absolute score of the best sent solution, and A — absolute score of participant's solution, and $[X]$ — an integer part of X (rounded down). In particular, the formula shows us, that if a participant has submitted the best solution, he will get 100 points.

According to the competition rules, the absolute score of the best solution is not reported until the end of the round. However, each participant get absolute and relative score of his own solution. Bob sent solution with absolute score A and received R points (relative score). Help Bob to get the minimum and maximum possible values for the absolute score of the best solution.

Input

The single line contains two positive integers — A and R , $1 \leq A \leq 10^9$, $1 \leq R \leq 100$.

Output

If for any value of the absolute score of the best solution Bob couldn't get these scores, print -1 . Otherwise, you should print two numbers — the smallest and the biggest possible absolute score of the best solution.

Example

stdin	stdout
100 99	-1
100 50	75 75
100 10	19 20

Problem H. We Admit No Defeat

Input file: `stdin`
Output file: `stdout`
Time limit: `2 s`
Memory limit: `256 Mb`

Bob trains to play Dota 2. He now plays as a hero called Mirana. However, in this problem it doesn't matter — heroes will not use their special abilities. Bob is going to fight against the enemy hero. And of course, Bob will admit no defeat! The battle will be honest, so the characters will simply hit each other. We may assume that heroes have the same attack speed and strike each other simultaneously. Heroes will strike each other, until one of them (or both) will not perish. Despite the overwhelming desire to win, the heroes are still under the influence of game mechanics. Here are the list of important parameters of the fight:

- Hero's health points
- Base damage
- Current armor

Damage which hero received can be calculated by the formula $D = BD / (1 + A * 0.06)$ — damage dealt to the hero, BD — base damage of the hero who has hit, A — armor of the hero, who received damage. The total value of damage is not rounded and is subtracted from the amount of health points the hero which get damage. If the amount of health points due to enemy strike will be non-positive, it is considered that the hero is killed. Since the two heroes attacking simultaneously, it may happen a situation where both heroes are killed. Base damage of each hero is strictly positive, so the battle will end.

In addition, heroes can buy items that increase one (or more) characteristics. Each hero can buy a maximum of 6 items, each item can be bought several times. Once items purchased, the battle begins. Help Bob to determine whether he can survive in the battle?

Input

The first line contains three numbers — the initial amount of hit points, base damage and armor of Bob's hero. The second line is a description of an enemy hero in the same format. The next line is a unique number N ($1 \leq N \leq 20$) — the number of possible items to purchase. Next lines will contain description of items. The first line contains the name of an item (a non-empty string of no more than 50 letters of the alphabet) and the number K ($1 \leq K \leq 3$) — number of bonuses from the subject. The next K lines is a description of bonuses. Each description of a bonus has a following format:

<BonusName> +X

where <BonusName> — type of changed characteristics:

- 'HP' — for health points of the hero
- 'Damage' — for base damage
- 'Armor' — for armor

X — the numerical value of the bonus to the characteristic. All bonuses from one subject have different <BonusName>. All characteristics (the initial characteristics of the characters and bonus items) — non-negative integers not exceeding 1000. Bonuses from items cannot be zero. Health points of hero is positive. All the names of items are different. See example below for the format of the input data.

Output

If for any set of items purchased by enemy hero, there is a set which Bob can buy to remain alive, print 1. If there is a set of items for enemy hero that Bob dies with any items, print -1.

Example

stdin	stdout
600 40 1 450 40 8 1 IronBranch 2 Damage +1 HP +19	-1
600 40 1 450 40 8 2 IronBranch 2 Damage +1 HP +19 Chainmail 1 Armor +5	1

Problem I. Wex-Wex-Wex

Input file: `stdin`
Output file: `stdout`
Time limit: `2 s`
Memory limit: `256 Mb`

Bob is playing Dota 2. One of the most complex and interesting hero in the game is Invoker. Like any other hero, Invoker was born for kill! There are N creep (noncontrollable units) that are lined up in a row in front of the hero, each one has exactly H Health. Creeps are placed with coordinates from 1 to N , one creep per every integer point. Invoker can kill creeps with his spells. In fact, he can use up to 10 different spells, but in this case he will only use 3 of them.

- Chaos Meteor — deals some fixed damage to all creeps on some interval.
- Sun Strike — base inflict damage is fixed, but it is divided equally among all *live* creeps on some interval. If damage is obtained by dividing is noninteger, then it is rounded down to the nearest integer.
- Tornado — raises creeps into the air for a while, making them immune to spells, and then drops it down to the ground and deals some fixed damage to each of the raised creeps. As any other spell, it works on some interval.

Creep is dead if the number of his health points after a spell was not greater than 0. Raised in the air with Tornado creeps are not considered as live when calculating damage of Sun Strike. If Tornado completes its action in time T , and in the same time another spell is used, first damage dealt by Tornado, and then begins the action of another spell.

Given by a set of spells determine the total amount of damage dealt by Invoker to creeps.

Input

The first line contains three positive integers N — number of creeps and H — the initial number of health points of creeps, and M — quantity of applied spells. $1 \leq N, M \leq 100\,000$, $1 \leq H \leq 1\,000\,000\,000$.

The next M rows describes spells, one per line in the order of their use of a hero. Spell description looks as follows:

`<Type> <Time> <L> <R> <D> [<T>]`, where

`<Type>` — type of spell casted (1 — Chaos Meteor, 2 — Sun Strike, 3 — Tornado);

`<Time>` — time needed to cast a spell;

`<L>` — left boundary of the area of spell (inclusive);

`<R>` — right boundary of the area of spell (inclusive);

`<D>` — damage of spell (damage from Sun Strike is divided between all living creeps);

`<T>` — spell duration (only for Tornado);

$1 \leq \text{Time}, T \leq 1\,000\,000\,000$; $1 \leq L \leq R \leq N$; $1 \leq D \leq 1\,000\,000\,000$.

Spells affect all the creeps on the segment from `<L>` to `<R>` inclusive. All time `<Time>` variables are different. It is guaranteed that at any time not more than one Tornado is active.

Output

Print a single number — the total damage dealt to creeps by the hero.

Example

stdin	stdout
5 10 7 1 1 2 5 4 1 2 2 2 8 2 3 1 3 9 3 4 1 4 2 2 2 5 3 5 4 2 6 3 5 4 1 7 5 5 6	44

Problem J. Warlock-2 (High)

Input file: `stdin`
Output file: `stdout`
Time limit: 2 s
Memory limit: 256 Mb

Bob trains to play Dota 2. Now he plays as a hero called Warlock. One of the abilities of the hero is Fatal Bounds — dark power «links» several enemy units together, causing them to receive additional damage. Spell works as follows: if one of the linked units take damage (as primary damage), all other units receive P percent of the unit received damage (as extra damage). Damage from link does not causes additional damage that was taken from that spell (extra damage does not cause additional damage to other units). Unit, that received base damage, does not receive additional damage. If unit dies while receiving basic damage (health points becomes non-positive), the additional damage will be calculated on the amount of health points of the unit before getting the main damage instead of the original values of incoming damage.

Bob wants to kill an enemy hero, so he linked him with N creeps using Fatal Bounds. Next we assume that the enemy hero doesn't get the base damage, while all the creeps will fight with Bob until their death. At this time some damage goes to Bob as additional damage. Of course, the creeps themselves will also suffer from the receipt of additional damage from Fatal Bounds. Bob can attack creeps arbitrary number of times, and each time he chooses one living creep, which will take base damage and the amount of damage (a positive real). Bob attacking creeps until they are killed. Knowing the initial amount of health points of creeps, determine the smallest and the biggest damage which enemy hero can get.

Input

The first line of the input file contains two integers: N — number of associated creeps with an enemy hero, $1 \leq N \leq 100000$ and P — percentage of additional damage relative to the basic damage, $0 \leq P \leq 100$. The second line contains N of positive integers not exceeding 1000 — the health points of creeps linked by Fatal Bounds.

Output

Output two real number with absolute or relative error of less than 10^{-6} — the smallest and the biggest damage which enemy hero can get.

Example

stdin	stdout
2 20 100 10	20.0000000000 21.6000000000

Problem K. World of Dota: Cross 2 (High)

Input file: `stdin`
Output file: `stdout`
Time limit: `2 s`
Memory limit: `256 Mb`

Bob is playing Dota 2, but he was tired of playing every time on the same map. So he decided to make a new map for this game. For simplicity, we assume that the map is a rectangle $N \times N$, each cell of which may be either free or busy (by tree or water). Bob wants that the enemy team (which consist of five characters) couldn't be grouped too close to one place. So, there shouldn't be available empty place to form a cross with a width and a height of 3 cells on the map.

```
.*.
***
.*.
```

In order to prevent the occurrence of such crosses, Bob can plant trees on the free cells. Furthermore, there are water which flows along the diagonal of the map. It means that each cell of the main diagonal of the source rectangle $N \times N$ is occupied by water. You cannot plant trees on a water, but also enemy heroes cannot stand up on it.

Bob wants to leave as much free cells to maneuver as he can, so the total count of trees should be as little as possible. Bob has already placed some trees on the map. Help Bob to place more trees as little as possible, so that five enemy heroes couldn't form a cross.

Input

The first line contains only a natural number — N — size of the map ($1 \leq N \leq 17$). Each of the next N lines contains N characters — description of the map. The character '.' corresponds to a free cell, the '#' — busy (by tree or water). It is guaranteed that all the cells of the main diagonal are busy.

Output

You should print the description of the new map — N lines of N characters each. If the cell should remain free, you must print the character '.' If the cell was originally occupied — '#' character, if it is initially empty cell where you want to add a tree — character '+'. If there are more than one optimal solution, you can print any of them.

Example

stdin	stdout
6 #..... .#..... ..#.... ...#...#..#	#..... .#...+. ..#... ...#.. .+...#.#
6 #..... .#..... ..#.... ...#... .#...#.#	#..... .#...+. ..#... ...#.. .#...#.#

Problem L. Wild Card: Plane (High)

Input file: **stdin**
Output file: **stdout**
Time limit: **2 s**
Memory limit: **256 Mb**

Bob has successfully reached the railway station, took the aeroexpress and arrived at the airport. Now Bob is ready to fly to the Starladder 2014 finals. However, another unpleasant event could hinder Bob from coming to the final in time. An extinct volcano suddenly woke up and began throwing flows of ash and smoke into the air. A few planes have already fallen in the immediate vicinity of the volcano, because they lost control due to the volcanic rocks, caught up in the air. To be safe the plane always has to be at least r far (along the surface of the earth) from the volcano. Thus, the segment of the airspace shaped as a circle with the center at the location of the volcano is closed to flights. Assuming that the Earth is a sphere of radius R , and also knowing the coordinates of the volcano, starting and ending points of the flight, determine the minimum distance the plane has to pass to safely carry out the flight. Altitude of the plane relative to the Earth has to be neglected — we consider the plane is flying at a low altitude along the surface.

Input

The first line contains two integers — R and r — radius of the Earth and the distance from the volcano to the border of restricted area. $1 \leq r, R \leq 10^9$, it is guaranteed that the area of the closed zone does not exceed a half of the Earth's surface area. The second line contains the coordinates of the volcano — Lat_v and $Long_v$ — latitude and longitude in degrees ($-90 \leq Lat_v \leq 90$, $-180 \leq Long_v \leq 180$). The third and fourth lines contain coordinates of the starting and ending points of the flight in the same format. Coordinates of all points are integers. It is guaranteed that takeoff and landing points differ and do not lie within or on the boundary of the restricted area.

Output

Print a single number — the smallest length of a safe path along the Earth surface.

Example

stdin	stdout
10 1 0 0 0 90 90 0	15.7079632679
10 1 0 0 0 45 0 -45	15.8083481818