Problem A. Ball Game

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	256 megabytes

A kindergarten teacher Natalia Pavlovna has invented a new ball game. This game not only develops the children's physique, but also teaches them how to count.

The game goes as follows. Kids stand in circle. Let's agree to think of the children as numbered with numbers from 1 to n clockwise and the child number 1 is holding the ball. First the first child throws the ball to the next one clockwise, i.e. to the child number 2. Then the child number 2 throws the ball to the next but one child, i.e. to the child number 4, then the fourth child throws the ball to the child that stands two children away from him, i.e. to the child number 7, then the ball is thrown to the child who stands 3 children away from the child number 7, then the ball is thrown to the child who stands 4 children away from the last one, and so on. It should be mentioned that when a ball is thrown it may pass the beginning of the circle. For example, if $n = 5$, then after the third throw the child number 2 has the ball again. Overall, $n - 1$ throws are made, and the game ends.

The problem is that not all the children get the ball during the game. If a child doesn't get the ball, he gets very upset and cries until Natalia Pavlovna gives him a candy. That's why Natalia Pavlovna asks you to help her to identify the numbers of the children who will get the ball after each throw.

Input

The first line contains integer n ($2 \le n \le 100$) which indicates the number of kids in the circle.

Output

In the single line print $n - 1$ numbers which are the numbers of children who will get the ball after each throw. Separate the numbers by spaces.

Examples

stdin	stdout
`10`	`2 4 7 1 6 2 9 7 6`
`3`	`2 1`

Problem B. T-shirts from Sponsor

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	256 megabytes

One day a well-known sponsor of a well-known contest decided to give every participant of the contest a T-shirt as a present. A natural problem occurred: on the one hand, it is not clear how many T-shirts of what sizes should be ordered, and on the other hand, one doesn't want to order too many T-shirts (and we do not exactly paper the walls with the oversupply). After considerable brain racking and some pre-estimating, the sponsor representatives ordered a certain number of T-shirts of sizes S, M, L, XL and XXL. The T-shirts turned out to bring good luck, that's why on the contest day there built up a line of K participants willing to get one. Every contestant is characterized by his/her desired T-shirt size (so it happens that for all the participants it is also one of the sizes S, M, L, XL and XXL). The participants come up to get a T-shirt one by one and try to choose the most suitable one, choosing it like this. If there is still a T-shirt of the optimal size left, that he/she takes it without further ado. Otherwise the contestant would prefer to choose a T-shirt with the size as close to the optimal one as possible (the distance between neighboring sizes is considered equal to one). If the variant of choice is not unique, the contestant will take a T-shirt of a bigger size (in case he/she grows more). For example, for a person whose optimal size is L the preference list looks like this: L, XL, M, XXL, S. Using the data on how many T-shirts of every size had been ordered by the organizers, on the size of contestants in the line determine who got a T-shirt of what size.

Input

The first line contains five non-negative integers $N_S, N_M, N_L, N_{XL}, N_{XXL}$ not exceeding 1000 which represent the number of T-shirts of the corresponding sizes. The second line contains an integer K ($1 \le K \le 1000$) which represents the number of participants. The next K lines contain the optimal T-shirt sizes for the contestants. The sizes are given in the order in which the participants stand in the line. It is guaranteed that $N_S + N_M + N_L + N_{XL} + N_{XXL} \ge K$.

Output

For each contestant, print a line containing the size of the T-shirt he/she got.

Examples

stdin	stdout
1 0 2 0 1 3 XL XXL M	XXL L L

Problem C. Hamsters and Tigers

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	256 megabytes

Today there is going to be an unusual performance at the circus — hamsters and tigers will perform together! All of them stand in circle along the arena edge and now the trainer faces a difficult task: he wants to swap the animals' positions so that all the hamsters stood together and all the tigers also stood together. The trainer swaps the animals in pairs not to create a mess. He orders two animals to step out of the circle and swap places. As hamsters feel highly uncomfortable when tigers are nearby as well as tigers get nervous when there's so much potential prey around (consisting not only of hamsters but also of yummier spectators), the trainer wants to spend as little time as possible moving the animals, i.e. he wants to achieve it with the minimal number of swaps. Your task is to help him.

Input

The first line contains number n ($2 \le n \le 1000$) which indicates the total number of animals in the arena. The second line contains the description of the animals' positions. The line consists of n symbols "H" and "T". The "H"s correspond to hamsters and the "T"s correspond to tigers. It is guaranteed that at least one hamster and one tiger are present on the arena. The animals are given in the order in which they are located circle-wise, in addition, the last animal stands near the first one.

Output

Print the single number which is the minimal number of swaps that let the trainer to achieve his goal.

Examples

stdin	stdout
3 HTH	0
9 HTHTHTHHT	2

Note

In the first example we shouldn't move anybody because the animals of each species already stand apart from the other species. In the second example you may swap, for example, the tiger in position 2 with the hamster in position 5 and then — the tiger in position 9 with the hamster in position 7.

Problem D. Parking Lot

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	256 megabytes

Nowadays it is becoming increasingly difficult to park a car in cities successfully. Let's imagine a segment of a street as long as L meters along which a parking lot is located. Drivers should park their cars strictly parallel to the pavement on the right side of the street (remember that in the country the authors of the tasks come from the driving is right side!). Every driver when parking wants to leave for themselves some extra space to move their car freely, that's why a driver is looking for a place where the distance between his car and the one behind his will be no less than b meters and the distance between his car and the one in front of his will be no less than f meters (if there's no car behind then the car can be parked at the parking lot segment edge; the same is true for the case when there're no cars parked in front of the car). Let's introduce an axis of coordinates along the pavement. Let the parking lot begin at point 0 and end at point L. The drivers drive in the direction of the coordinates' increasing and look for the earliest place (with the smallest possible coordinate) where they can park the car. In case there's no such place, the driver drives on searching for his perfect peaceful haven. Sometimes some cars leave the street and free some space for parking. Considering that there never are two moving cars on a street at a time write a program that can use the data on the drivers, entering the street hoping to park there and the drivers leaving it, to model the process and determine a parking lot space for each car.

Input

The first line contains three integers L, b и f ($10 \le L \le 100000, 1 \le b, f \le 100$). The second line contains an integer n ($1 \le n \le 100$) that indicates the number of requests the program has got. Every request is described on a single line and is given by two numbers. The first number represents the request type. If the request type is equal to 1, then in that case the second number indicates the length of a car (in meters) that enters the street looking for a place to park. And if the request type is equal to 2, then the second number identifies the number of such a request (starting with 1) that the car whose arrival to the parking lot was described by a request with this number, leaves the parking lot. It is guaranteed that that car was parked at the moment the request of the 2 type was made. The lengths of cars are integers from 1 to 1000.

Output

For every request of the 1 type print number `-1` on the single line if the corresponding car couldn't find place to park along the street. Otherwise, print a single number equal to the distance between the back of the car in its parked position and the beginning of the parking lot zone.

Examples

stdin	stdout

30 1 2
6
1 5
1 4
1 5
2 2
1 5
1 4
	0
	6
	11
	17
	23

30 1 1
6
1 5
1 4
1 5
2 2
1 5
1 4
	0
	6
	11
	17
	6

10 1 1
1
1 12
	-1

Problem E. Comb

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	256 megabytes

Having endured all the hardships, Lara Croft finally found herself in a room with treasures. To her surprise she didn't find golden mountains there. Lara looked around and noticed on the floor a painted table $n \times m$ panels in size with integers written on the panels. There also was a huge number of stones lying by the wall. On the pillar near the table Lara found a guidance note which said that to get hold of the treasures one has to choose some non-zero number of the first panels in each row of the table and put stones on all those panels to push them down. After that she will receive a number of golden coins equal to the sum of numbers written on the chosen panels. Lara quickly made up her mind on how to arrange the stones and was about to start when she noticed an addition to the note in small font below. According to the addition, for the room ceiling not to crush and smash the adventurer, the chosen panels should form a comb. It was explained that the chosen panels form a comb when the sequence $c_1, c_2, \ldots, c_n$ made from the quantities of panels chosen in each table line satisfies the following property: $c_1 > c_2 < c_3 > c_4 < \ldots$, i.e. the inequation mark interchanges between the neighboring elements. Now Lara is bewildered and doesn't know what to do. Help her to determine the largest number of coins she can get and survive at the same time.

Input

The first line contains a pair of integers n, m ($2 \leq n, m \leq 1500$). Next n lines contain m integers each — that is the table itself. The absolute value of the numbers in the table does not exceed 10000.

Output

Print the single number — the maximum number of coins Lara can get.

Examples

stdin	stdout

2 2
-1 2
1 3
	2

Problem F. Hercule Poirot Problem

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	256 megabytes

Today you are to solve the problem even the famous Hercule Poirot can't cope with! That's why this crime has not yet been solved and this story was never included in Agatha Christie's detective story books.

You are not informed on what crime was committed, when and where the corpse was found and other details. We only know that the crime was committed in a house that has n rooms and m doors between the pairs of rooms. The house residents are very suspicious, that's why all the doors can be locked with keys and all the keys are different. According to the provided evidence on Thursday night all the doors in the house were locked, and it is known in what rooms were the residents, and what kind of keys had any one of them. The same is known for the Friday night, when all the doors were also locked. On Friday it was raining heavily, that's why nobody left the house and nobody entered it. During the day the house residents could

- open and close doors to the neighboring rooms using the keys at their disposal (every door can be opened and closed from each side);
- move freely from a room to a room if a corresponding door is open;
- give keys to one another, being in one room.

"Little grey matter" of Hercule Poirot are not capable of coping with such amount of information. Find out if the positions of people and keys on the Thursday night could result in the positions on Friday night, otherwise somebody among the witnesses is surely lying.

Input

The first line contains three preset integers n, m и k ($1 \le n, m, k \le 1000$) — the number of rooms, the number of doors and the number of house residents respectively. The next m lines contain pairs of room numbers which join the doors. The rooms are numbered with integers from 1 to n. There cannot be more that one door between the pair of rooms. No door connects a room with itself. The next k lines describe the residents' position on the first night. Every line contains a resident's name (a non-empty line consisting of no more than 10 Latin letters), then after a space follows the room number, then, after a space — the number of keys the resident has. Then follow written space-separated numbers of the doors that can be unlocked by these keys. The doors are numbered with integers from 1 to m in the order in which they are described in the input data. All the residents have different names, uppercase and lowercase letters considered to be different. Every m keys occurs exactly once in the description. Multiple people may be present in one room, some rooms may be empty. The next k lines describe the position of the residents on the second night in the very same format. It is guaranteed that in the second night's description the residents' names remain the same and every m keys occurs exactly once.

Output

Print "YES" (without quotes) if the second arrangement can result from the first one, otherwise, print "NO".

Examples

stdin	stdout

`2 1 2`
`1 2`
`Dmitry 1 1 1`
`Natalia 2 0`
`Natalia 1 1 1`
`Dmitry 2 0`
	`YES`

`4 4 3`
`1 3`
`1 2`
`2 3`
`3 4`
`Artem 1 1 4`
`Dmitry 1 1 2`
`Edvard 4 2 1 3`
`Artem 2 0`
`Dmitry 1 0`
`Edvard 4 4 1 2 3 4`
	`NO`

Problem G. Emperor's Problem

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	256 megabytes

It happened at the times of the Great Berland Empire. Once the Emperor dreamt that the Messenger from the gods ordered to build a temple whose base would be a convex polygon with n angles. Next morning the Emperor gave the command to build a temple whose base was a regular polygon with n angles. The temple was built but soon the Empire was shaken with disasters and crop failures. After an earthquake destroyed the temple, the Emperor understood that he somehow caused the wrath of gods to fall on his people. He ordered to bring the wise man. When the wise man appeared, the Emperor retold the dream to him and asked "Oh the wisest among the wisest, tell me how could I have infuriated the Gods?". "My Lord," the wise man answered. "As far as I can judge, the gods are angry because you were too haste to fulfill their order and didn't listen to the end of the message".

Indeed, on the following night the Messenger appeared again. He reproached the Emperor for having chosen an imperfect shape for the temple. "But what shape can be more perfect than a regular polygon!?" cried the Emperor in his dream. To that the Messenger gave a complete and thorough reply.

- All the vertices of the polygon should be positioned in the lattice points.
- All the lengths of its sides should be different.
- From the possible range of such polygons a polygon which maximum side is minimal possible must be chosen.

You are an obedient architect who is going to make the temple's plan. Note that the polygon should be simple (having a border without self-intersections and overlapping) and convex, however, it is acceptable for three consecutive vertices to lie on the same line.

Input

The first line contains the single number n ($3 \le n \le 10000$).

Output

Print "YES" (without quotes) in the first line if it is possible to build a polygon possessing the needed qualities. In the next n lines print integer coordinates of the polygon vertices in the order in which they would be passed counter clockwise. The absolute value of the coordinates shouldn't exceed 10^9. No two vertices can coincide. It is permitted to print any of the possible solutions. Print "NO" if to build the polygon is impossible.

Examples

stdin	stdout

`3`	`YES`
	`0 0`
	`1 0`
	`0 2`

`3`	`YES`
	`0 1`
	`-1 0`
	`-1 -1`