Problem A. FIFA World Cup

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

The Qatar World Cup has begun! Many countries from all over the world are participating in this tournament.

We know that before the start of each match many bettors bet on the team they like or think will win. Suppose that two teams are bet on in a match in the ratio $x : y$. Then if the first team wins, all bettors bet on it get $\frac{y}{x}$ times the profit, and if the second wins, all bettors bet on it get $\frac{x}{y}$ times the profit.

Mr. He is a veteran fan. Before the match starts, he scores all the n teams. A total of m games will be played. What a surprise, the ratio of the amount of money bet on each game is exactly equal to the ratio of his scores on the two teams. Now, Vingying also wants to bet 1 on one of the teams (this 1 has a negligible effect on the ratio of bets) and he wants to know the maximum amount of money he can get.

Note: To simplify the question, the possibility of a tie is not considered in this question.

Input

The first line contains an integer $n(2 \le n \le 100)$, denoting the number of teams.

Each of the next n lines contains a string $Name_i$ and an integer $Score_i(1 \le Score_i \le 200)$, denoting the name of the i-th team and its score. The string contains only English letters, and the length won't exceed 20.

The next line contains an integer $m(1 \le m \le 100)$, denoting the number of matches.

Each of the next m lines contains two strings A_i and B_i, denoting that the i-th match is between the team A_i and the team B_i. It is guaranteed that $A_i \ne B_i$ and both of them appear in the set of $Name_i$.

Output

For each match, print a real number in one line, denoting the maximum amount of money that Vingying can get.

Your answer will be considered correct if and only if the absolute or relative error between yours and the jury is not bigger than 10^{-6}.

Example

<table><tr><th>standard input</th><th>standard output</th></tr><tr><td><pre>4
Japan 1
Germany 100
SaudiArabia 10
Argentina 150
4
Japan Germany
SaudiArabia Argentina
Germany Argentina
SaudiArabia Japan</pre></td><td><pre>100.0
15.0
1.5
10.0</pre></td></tr></table>

Problem B. Calculate probability

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	256 megabytes

Batman is caught by the Joker, but Joker does not want to kill Batman directly, he wants to play a game with Batman. Now Batman has a non-decreasing array A length of n. Joker has a non-decreasing array B length of m. Joker asks Batman to choose a number a randomly from array A. Meanwhile he chooses a number b randomly from array B. If $a > b$, Joker will let Batman go. Batman now wants to know how many scenarios he could survive.

Input

The first line contains two integers n and m $(1 \le n, m \le 10^6)$

The second line contains n integers, $a_1, a_2 ... a_n$ $(0 \le a_1 \le ... \le a_i \le ... \le a_n \le 10^6)$,where a_i is the ith integer in array A

The third line contains m integers, $b_1, b_2 ... b_m$ $(0 \le b_1 \le ... \le b_i \le ... \le b_m \le 10^6)$,where b_i is the ith integer in array B

Output

Print one integer — the number of the scenarios he could survive.

Example

standard input	standard output

`5 5`
`2 3 4 5 6`
`1 2 3 4 5`
	`15`

Note

The answer may exceed the **int** range.

Problem C. Extended Monty Hall Problem

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

There're n doors on the stage, exactly one car is behind one of the doors. A game with the above setting is played on the stage.

The game has m turns. At the beginning of each turn, player `Antimo51` is supposed to choose one of the closing doors **without opening it** and the host will open some doors **other than that door**. Specifically, in the i-th turn of the game, the host will open k_i doors that are not opened, not chosen, and not in front of the car. **Note that a door will not be closed again once it is opened.** At the end of each turn, `Antimo51` can choose to terminate the game. If so, he can choose a door (can be the same as one of those chosen in the previous turns) and get the prize behind it. If all m turns are played, `Antimo51` must choose a door and get the prize behind it.

The host will try to minimize `Antimo51`'s probability of getting the car. Please find out a strategy for `Antimo51` that maximizes that probability. Output the maximum probability that you can get.

Input

The first line contains two integers n and m ($3 \le n \le 100, 1 \le m \le n-2$), denoting the number of doors and the number of turns.

The second line contains m integers, where the i-th integer denotes k_i ($1 \le k_i \le n-2$), the number of the doors to be chosen by the host in the i-th turn.

It's guaranteed that $\sum k_i \le n-2$.

Output

Output the answer as a minimal fraction.

Specifically, Output two integers a and b separated by a space, denoting that the answer equals $\frac{a}{b}$. The numbers a and b must satisfy that $\gcd(a,b)=1$ and $a,b>0$, where $\gcd(a,b)$ denotes the greatest common divisor of a and b. **Note that, when the answer is an integer, although $b=1$, you should still output b.**

Examples

standard input	standard output

`3 1`
`1`
	`2 3`

`4 1`
`2`
	`3 4`

`6 2`
`2 1`
	`5 12`

Note

In the first example, `Antimo51` should choose the door other than the one chosen at the beginning, resulting in a probability of $\frac{2}{3}$.

Problem D. Schedule Planning

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

In the year of 114514, the annual *UESTC Programming Competition of Interest* is about to kick off. As the number of participants is increasing explosively, for example, last year, 1919810 students signed up for the competition. In order to make the competition more exciting, the Organizing Committee decides to change the competition system to the elimination competition.

The rules of the elimination competition are as follows: Suppose there are 2^n contestants, arranged in the order from 1 to 2^n, and the relative position of the contestants will not change during the competition. There are a total of n rounds in the competition. At the first round, contestants numbered $2i - 1$ and contestants numbered $2i$ will solve a problem by programming within a limited time. The contestants who pass with less time will be promoted. The contestants who do not advance will be eliminated directly, and there will be no tie. At the second round, two contestants numbered from $4i - 3$ to $4i$ who won the last round will continue to compete under the same rules... At the $j - th$ round, two contestants numbered from $(i - 1)2^j + 1$ to $i2^j$ will compete until there is only one contestant left, who is the champion of the whole competition and will win the prize of unique sleeping black tea!

Now, the committee has entrusted *Bob* with the task of arranging the number of contestants. However, *Bob* is busy eating snow, so he gives this tough task to you and explains the conditions of the contestants. First of all, to ensure that the competition can continue, the number of participants must be 2^n; Secondly, each contestant will have an ability value between 1 and 2^n.**And the ability values between any two contestants are different**. All the contestants can solve all the questions. The faster the contestants with higher ability value can solve the questions. For example, the contestants with the ability value of 2 and the ability value of 3 have both worked out a question, but the contestant with the ability value of 3 can work faster, so the contestant with the ability value of 3 is promoted, and the contestant with the ability value of 2 is eliminated.

But *Bob* has a good friend. He hopes you can plan a way to arrange the number of contestants so that this friend can promote as many times as possible.

Input

The first line contains two integers $n, m (0 \le n < 63, 1 \le m \le 2 \times 10^5)$, representing the number of contestants is 2^n and the number of inquiries.

The next m lines, each with an integer $x_i (1 \le x_i \le 2^n)$, indicating that *Bob* wants the contestant whose capability value is x_i to be promoted as many times as possible. All queries are independent with each other, which means that you can rearrange the number of contestants for each query, and you only need to ensure that in the current query the contestant whose capability value is x_i be promoted as many times as possible without considering other contestants.

Output

A total of m lines, for each line with an integer y_i, which means the contestant whose capability value is x_i can be promoted at most y_i times.**You do not need to output the concrete arrangement**.

Example

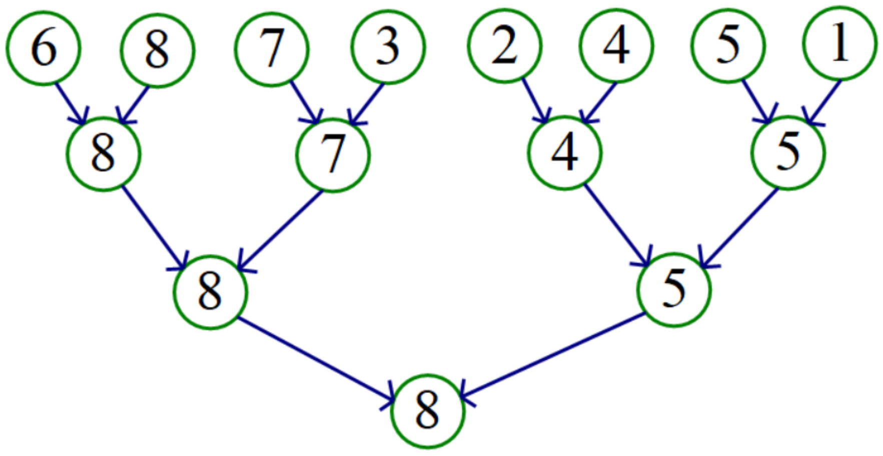
standard input	standard output

3 3
1
5
8
	0
	2
	3

Note

For question 1, no matter how to arrange the number, the contestant with the ability value of 1 must be eliminated in the first round, so the number of promotion is 0;

For question 2, one arrangement is 68732451. The flow chart of the competition can be made as follows:



Therefore, the promotion times are 2;

For question 3, no matter how to arrange the number, the contestant with the ability value of 8 must be the champion of this competition. The number of promotions is 3, and the this person will win the prize of unique sleeping black tea.

Problem E. Exchange Gifts

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Christmas is coming. Now n kids in the Xifeng Huhu Welfare Home are going to exchange gifts.

Before that, every kid has prepared a gift and so there are n gifts in total. For convenience, we number the kids from 1 to n, and the gift prepared by the i-th kid is also numbered i $(i = 1, 2, ..., n)$.

Initially, every kid will have a gift randomly selected from all of the gifts, and then they sit in a circle around the Christmas tree in the order of number 1 to n clockwise.(It means, kids with number 1 and n are adjacent.) Then they will start to exchange gifts. Each time for exchange, every kid will give the gift in his hands to the kid sitting in his left, and take the gift given by the kid sitting in his right. To be specific, assuming that after k times of exchange, the number of the gift in the i-th kid's hands is $a_{k,i}$, then after the $(k+1)$-th exchange, $a_{k+1,i} = a_{k,i-1}$. Specially, $a_{k+1,1} = a_{k,n}$.

For example, if there are three kids, and the number of the gifts in their hands initially are $2, 1, 3$ respectively, then after the first time of exchange, the number of the gifts in their hands will become $3, 2, 1$ respectively.

However, kids don't want to get gifts prepared by themselves, so they wonder how many times they should exchange their gifts **at least** (possibly 0), so that every kid will have a gift **not** prepared by himself. That is, for every i from 1 to n, $a_i \neq i$ holds. Here, a_i represents the number of the gift in the i-th kid's hands.

Input

The input consists of multiple test cases. The first line contains a single integer t $(1 \leq t \leq 10^5)$ — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n $(2 \leq n \leq 2 \cdot 10^5)$ — the number of the kids.

The second line of each test case contains n integers $a_1, a_2, ..., a_n$ $(1 \leq a_i \leq n)$ — the number of the gift in the i-th kid's hands initially. It's guaranteed that every number from 1 to n appears exactly once.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case print a line containing a single integer k , denoting that at least k times of exchange should be performed in order to achieve the requirement.

If such k doesn't exist, print -1 .

Examples

standard input	standard output

2
3
2 1 3
5
1 5 4 2 3
	-1
	2

3
5
2 3 5 1 4
7
4 2 6 3 1 5 7
10
7 3 1 6 2 4 10 8 9 5
	0
	1
	4

Problem F. The Circle Game

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

And the seasons they go round and round
And the painted ponies go up and down
We're captive on the carousel of time
We can't return, we can only look behind from where we came
And go round and round and round in the circle game...

— Joni Mitchell, *The Circle Game*

One day, vv123 thought of the days when he played circle game with his classmates in PE class, when he was just an elementary school student.

There are n(n is even) kids forming a circle hand by hand, the strength of the n kids are $1, 2, \dots, n$ respectively. vv123 wants to rearrange the kids in the circle, so that the circle is firmest.

Suppose that the villain can only attack half the circle at a time, and it will obviously attack the weakest part of the circle. So, we define the firmness of the circle as the smallest sum of any $\frac{n}{2}$ consecutive kids' strength.

For the given n, what's the maximum firmness that can be achieved?

Input

An integer n. ($n = 2k, 2 \leq n \leq 10000$)

Output

An integer, denoting the maximum firmness that can be achieved.

Examples

standard input	standard output
`4`	`4`
`6`	`10`

Note

In the first example, $n = 4$, there are 6 ways to form the circle:

$1 - 2 - 3 - 4 - 1$, the firmness is $\min(1 + 2, 2 + 3, 3 + 4, 4 + 1) = 3$.

$1 - 2 - 4 - 3 - 1$, the firmness is $\min(1 + 2, 2 + 4, 4 + 3, 3 + 1) = 3$.

$1 - 3 - 2 - 4 - 1$, the firmness is $\min(1 + 3, 3 + 2, 2 + 4, 4 + 1) = 4$.

$1 - 3 - 4 - 2 - 1$, the firmness is $\min(1 + 3, 3 + 4, 4 + 2, 2 + 1) = 3$.

$1 - 4 - 2 - 3 - 1$, the firmness is $\min(1 + 4, 4 + 2, 2 + 3, 3 + 1) = 4$.

$1 - 4 - 3 - 2 - 1$, the firmness is $\min(1 + 4, 4 + 3, 3 + 2, 2 + 1) = 3$.

So, the maximum firmness that can be achieved is 4.

In the second example, there are 120 ways to form the circle. The maximum firmness that can be achieved is 10, and one possible way to achieve this is $1 - 4 - 5 - 2 - 3 - 6 - 1$.

Problem G. Arrange the desktop

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Charming is a slightly obsessive college student. One day, bug appears on his phone and the icons on his mobile desktop are out of order. Help *Charming* arrange the mobile desktop **in order**!

Specifically, the mobile desktop is a grid of $n \cdot m$, with rows 1, 2 ... n from top to bottom, columns 1, 2, ... m from left to right. each icon occupies a grid in the grid diagram.

In order means that all icons are arranged row by row, starting from the first row. That is, for any icon, its position (i, j) in the grid satisfies: $(i-1) \cdot m + j \le k$, and k is the number of icons on the desktop.

When an icon moves from (x_1, y_1) to (x_2, y_2), it can be seen as moving $|x_1 - x_2| + |y_1 - y_2|$ distance.

Now *Charming* wants to know, at least how far does it have to move to put the table in order? (Note: only one icon can be placed in the same square. You can consider that icons can only be moved to the position where there is no icon.)

Input

The first line contains three integers n, m and k, means that *Charming*'s mobile desktop have n rows and m columns with k icons. $(1 \le n, m \le 500, 0 \le k \le n \cdot m)$

Each of the next n lines contain a 01 string with length of m.

If the character in the $i+1$ line and j column is 1, it means that there is an icon in the i row and j column of the mobile desktop.

Output

Print an integer – the least distance *Charming* need to move to arrange the mobile desktop in order.

Examples

standard input	standard output

`1 1 0`
`0`
	`0`

`3 2 1`
`00`
`10`
`00`
	`1`

`4 4 8`
`0110`
`0101`
`0001`
`1110`
	`10`

Note

In the third test case, you can move like this. It can be shown that 10 is the least distance to arrange the mobile desktop in order.

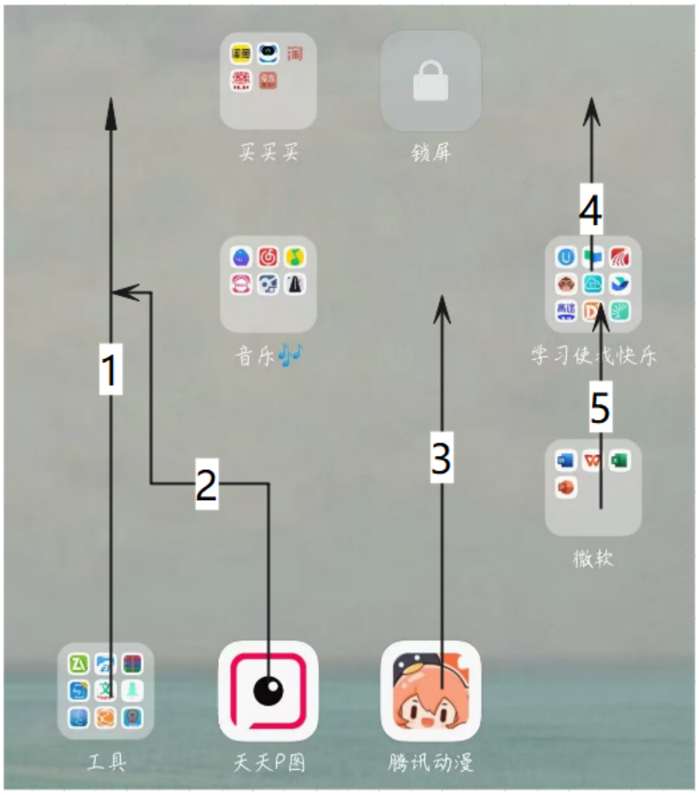


Figure for test3