Problem A: Chop Chop

Filename: `chopchop`
Time limit: `2 seconds`

Eric has recently started a healthy lifestyle change and is preparing his meals for the rest of the week. Eric has all the vegetables laid out on a cutting board ready to be chopped. He must choose some contiguous subarray of the vegetables to form a recipe. However, he enjoys a recipe if and only if there is at least one vegetable in that recipe appearing only once. Help Eric check that no matter what subarray he chooses the recipe will be enjoyed.

Input

The input begins with an integer ***T*** ($1 \le T \le 20$), the number of test cases. For each test case, the first line contains a single integer ***n*** the number of vegetables on the cutting board. On the following line is ***n*** space separated integers, a_i ($0 \le a_i \le 10^9$), the ID of each vegetable.

The scoring for this problem will be a little different. There will be only two input batches, one small and one large.

- Small input file: ($1 \le n \le 10^3$) worth 20 points
- Large input file: ($1 \le n \le 10^5$) worth 80 points

Output

For each test case, output 1 if Eric will enjoy all recipes or 0 otherwise, on a line by itself.

Samples

Input	Output
4 5 1 2 3 4 5 5 1 1 1 1 1 5 1 2 3 2 1 5 1 1 2 1 1	1 0 1 0

Problem B: Memory

Filename: `memory`

Time limit: `8 seconds`

Arup loves to play the card game Memory with his children. For those who don't remember how memory works, here is a quick explanation of the game:

The game begins with a collection of cards all containing different images randomly distributed face down on a table. The images are all in pairs. For example, if there are 20 cards in the game, there are 10 different images each of which has two cards with that image. On a given player's turn, that player first turns over one of the remaining cards to reveal its image to both players. He/she then turns over another card. If the two images match, he removes those cards from the table, scores a point, and takes another turn. Otherwise, he flips both of the cards back over and the other player then takes a turn. The game ends when either all of the cards are gone or when a predetermined number of consecutive turns have gone by in which no matches were made. If that happens, the game ends and the players' final scores are what they had when the game ended. Although Arup's daughters tend to forget the location of the images making the game relatively easy for Arup to win, he has begun to wonder if both players had perfect memory and played perfect strategy, what is the expected difference in the scores between the two players. Your job is to calculate this for Arup.

If a player could at any time make a pair because he has either revealed a card that he knows its pair's location or because two cards revealed are the same, he will greedily take it and score the point and thus take another turn.

Input

The first and only line of input will contain two integers ***n*** and ***m*** ($0 \le n, m \le 300$). ***n*** represents the number of cards that are being used (***n*** is guaranteed to be even) and ***m*** represents the number of turns with no matches before the game ends by default.

Output

Output a single decimal value with 4 digits after the decimal point representing the average difference between the two players. For example, if it is expected that the first player will win by 2.23 points, output 2.2300. If the first player is expected to lose by 2.23 points, output -2.2300. Round your answer to four decimal places using the standard .5 up rounding.

Samples

Input	Output
`10 100`	`-0.1556`
`2 10`	`1.0000`
`20 20`	`-0.0355`

Problem C: Happy Digits

Filename: `happy`
Time limit: `4 seconds`

Some digits are happy and some are not. A digit's happiness is defined as whether the digit is ***strictly greater*** than its predecessor (e.g., the ten-digit's predecessor is the one-digit, and so on). Thus, in the number 1963, 1 is not happy, but 9 and 6 are. In both cases, the ones digit doesn't have any happiness. The happiness of single-digit numbers is undefined.

A happiness mask is a binary string. The ***i***th digit from the right is a ***1*** if the ***(i+1)***th digit in the corresponding number is happy. For example, ***1963*** matches a happiness mask of ***011*** because ***1 < 9***, ***9 > 6***, and ***6 > 3***. To compare two masks of unequal lengths, the shorter mask is padded with zeros on the left.

Your task is to do operations on a multiset ***S*** consisting of many numbers. You have to respond to three types of queries:

1. Put a new number into ***S***.
2. Remove all numbers matching a happiness mask ***M*** from ***S***.
3. Output how many numbers in ***S*** match a happiness mask ***S***.

Initially, ***S*** is empty. A multiset is an unordered set of items allowing duplicates.

Input

The first line will be an integer ***q*** ($1 \le q \le 10^5$), the number of queries to process. The next ***q*** lines each contain a query. Each line begins with a query code: 1, 2, or 3. Below is the input for each query:

1. Following the query code will be an integer ***n***, ($10 \le n < 10^{40}$) to insert into ***S***. This indicates that the integer n must be added to the multiset ***S.***
2. Following the query code is a happiness mask ***M*** represented as a binary string of up to thirty-nine characters, representing that all integers with mask ***M*** should be removed from ***S***.
3. Like in the second query, following the query code is a happiness mask ***M*** represented as a binary string of up to thirty-nine characters. This is an actual query, asking for the number of integers in ***S*** that have the happiness mask ***M***.

Output

Do not output anything for first- and second-type queries. Only output the answer to type-three queries. Print each answer on its own line.

Samples

Input	Output
4 1 1963 3 011 2 011 3 011	1 0
3 3 11 1 987 3 11	0 1

Problem D: No Squares Allowed

Filename: `nosquares`
Time limit: `2 seconds`

No fluff here: you are given an array ***a*** of ***N*** positive integers, followed by ***Q*** queries.

Answer the following question for each query: given a subarray of ***a***, what is the maximum number of elements you can select from that subarray such that no two elements multiply to form a perfect square?

Input

The first line of input contains two integers ***N*** and ***Q*** ($1 \le N, Q \le 10^5$). The following line will contain ***N*** integers a_i ($1 \le a_i \le 10^6$). The following ***Q*** lines will each contain two integers ***L*** and ***R*** ($1 \le L \le R \le N$), the indices of the leftmost and rightmost elements,inclusive, of the subarray for the i^{th} query.

Output

Print ***Q*** lines, the i^{th} line containing the answer to the i^{th} query.

Samples

Input	Output

`8 4`
`1 2 3 4 5 6 7 8`
`1 8`
`2 4`
`3 7`
`1 1`
	`6`
	`3`
	`5`
	`1`

`7 3`
`16 48 17 21 35 9 11`
`1 7`
`3 6`
`2 3`
	`6`
	`4`
	`2`

Problem E: Airport Shuttle

Filename: `airport`
Time limit: `2 seconds`

When all of the out of state campers arrived to Orlando International Airport, the SI@UCF staff had to make several runs to the airport to pick up all of the campers. Naturally, none of the staff members wanted to wait too long at the airport. Each staffer's wait time was simply the difference in arrival times between the first and last camper he/she picked up.

Luckily, all of the staffers have access to arbitrarily large shuttle buses! But, in addition to wanting to minimize their wait time at the airport, none of the staffers want to make more than one airport run.

Given the number of SI@UCF staffers, as well as the arrival times of each camper flying into Orlando International Airport, determine the minimum amount of time, T, in minutes, such that no staffer will have to wait more than T minutes.

Input

The first line of input contains two space separated positive integers: ***n*** (***n*** $\le 10^5$), and ***k*** (***k*** $\le$ ***n***), where ***n*** represents the number of campers flying into the airport and ***k*** represents the number of staffers. The second line of input contains the ***n*** arrival times, in minutes, separated by spaces, of each camper. Each of these arrival times will be positive integers less than or equal to 10^9.

Output

On a single line by itself, output the minimum number of minutes T, such that no staffer waits more than T minutes, no staffer takes more than one trip, and each camper gets picked up.

Samples

Input	Output
`9 3` `10 10 30 200 205 210 215 220 500`	`20`
`5 2` `10 2 16 19 5`	`8`

Problem F: Brian and Castle

Filename: `castle`
Time limit: `4 seconds`

Brian has been playing Minecraft for so long that he fell asleep at his keyboard. Now he is dreaming about building a Minecraft castle, and for some reason you're there to help him. Brian has built several towers around his world, and he wants you to build some bridges between them so that he can reach any tower from any other tower using only the bridges.

However, because Brian is a harsh taskmaster with very specific wishes, you can not build bridges between any pair of towers you want. Brian has given you a list of pairs of towers you can build between, as well as how long he thinks it would take to build a bridge between those towers. You want to spend as little time as possible building these bridges so you can get out of this dream and go back to practicing programming problems. Brian guarantees that there is at least one set of bridges you can choose to satisfy his demands. How quickly can you finish building Brian's bridges?

Input

The first line of input contains two integers N ($2 \le N \le 10^5$) and M, (N-1 $\le M \le 10^5$), the number of towers and the number of possible bridges, respectively. The following M lines each contain three integers, u_i, v_i ($1 \le u_i$, $v_i \le N$, $u_i \ne v_i$), and t_i ($1 \le t_i \le 10^6$), indicating that you can build a bridge between towers u_i and v_i in t_i minutes.

Output

On a line by itself, print a single number, the minimum number of minutes it will take you to connect all of the towers in Brian's castle.

Samples

Input	Output
3 3 1 2 5 2 3 7 1 3 4	9
4 5 1 2 6 2 3 8 3 4 2 1 4 5 2 4 7	13

Problem G: RPS

Filename: `rps`

Time limit: `1 second`

Eric is playing rock-paper-scissors. However, Eric does not like to lose. Calculate the probability of winning ***n*** games of rock-paper-scissors without losing. Ties do not count as a loss.

Input

The first and only line contains an integer ***n*** ($1 \le n \le 10^{18}$) the number of games to win without losses.

Output

The probability of winning ***n*** games without any losses can be represented as a fraction reduced to lowest terms, p/q, where p and q share no common factors (gcd(p, q) = 1). Output the remainder when p+q is divided by 1,000,000,007.

Samples

Input	Output
`1000`	`688423211`
`10000`	`905611806`

Problem H: Toll Roads

Filename: `tollroads`
Time limit: `4 seconds`

Eric is trying to get from his house to campus as fast as possible, but he hates paying for tolls. When tolls were first introduced, there was promise of them being removed once the cost of the road was collected. Eric doubts this will ever happen, so he wants to try and cheat the system and is only willing to pay up to X dollars in tolls in one trip.

Given a description of the city, where each road takes time t_i and cost c_i in tolls (potentially zero if there is no toll on this road), help Eric find the shortest amount of time needed to get from his house (node 1) to campus (node N). Each road takes the same amount of time and cost no matter which direction the road is traveled along.

Input

The first line of input contains three integers N, M, and X, ($1 \le N, M, X \le 10^3$), the number of nodes in the city, the number of roads, and the maximum amount of dollars Eric is willing to pay in tolls.

The following M lines contain four integers each, u_i, v_i ($1 \le u_i, v_i \le N$, $u_i \ne v_i$), c_i ($0 \le c_i \le 10^5$) and t_i ($0 \le t_i \le 10^5$), describing the start and end of each road as well as the cost (toll) for the road and the time it takes to travel along the road.

Output

Print a single line containing the fastest time Eric can get to campus without overspending. If it is not possible for Eric to get to campus without overspending, print -1.

Samples

Input	Output

`4 6 4`
`1 2 2 100`
`1 3 2 200`
`1 4 5 1`
`2 3 2 5`
`2 4 2 50`
`3 4 2 20`
	`150`

`2 1 1` `1 2 2 1`	`-1`