Problem A: Find My Arup

Filename: `arup`
Time limit: `2 seconds`

As everyone has an Arup, a brilliant idea for the next killer app is obviously one that can find your Arup. But how can you ensure that you will find your Arup and not someone else's Arup. Your Arup is special and you cannot settle for finding just any old Arup.

That is why the find my Arup app will feature special advanced *Arup Locating Technology™*. To help find your Arup, other Arup's will be used as *Telepathic Arup Beacons™*. Specifically, each Arup i, not your Arup, is currently in the position (x_i, y_i). Your Arup is in some position (x_a, y_a). Each Arup, not your Arup, will send the result of the following function to your phone:

$$d(i) = max(|x_i - x_a|, |y_i - y_a|)$$

From this information, it should be possible to locate your Arup precisely. If not, either two things have happened, there is not enough information to uniquely locate your Arup, or there is an issue with the *Arup Locating Technology™* and your Arup's location is not consistent with the information given. You should determine which is the case and print Arup's location if possible. (Note: Your app should also take into account that all Arups, even your Arup, will always stand at integer coordinates and no two Arups, even your Arup, will occupy the same position.)

Input

The first line of input will be a single integer ***n*** (1 ≤ ***n*** ≤ 100), representing the number of Arups, not including your Arup.

The next ***n*** lines contains three integers x_i, y_i, and ***d(i)*** (-100 ≤ x_i, y_i ≤ 100, 0 ≤ ***d(i)*** ≤ 100), representing the location of Arup ***i*** and his beacon function from above.

Output

If you can find your Arup, print his location as integer coordinates x_a y_a. If your Arup could be at multiple locations print "`Multiple`" without quotes. If a beacon Arup must be misreporting, print "`Error`" without quotes.

Samples

Input	Output
1 1 1 0	Error
1 1 1 1	Multiple
2 -1 -1 1 2 2 2	0 0

Problem B: Quadruple Chomp

Filename: `chomp4`
Time limit: `3 seconds`

Nom Chomsky the chain chomp regularly sneaks bites of food from Bowser's dinner table. All the food on the table is arranged on a line and Chomsky is able to eat any contiguous section in a single bite. After honing his technique he can impressively manage four bites before being caught. Chomsky knows how tasty each dish is wishes to maximize the combined tastiness of his four bites. However he must be careful as some dishes are so bad tasting that they have a negative tasty value! Find the maximum tastiness Chomsky can obtain with four or less bites.

Input

The first line of input contains a single positive integer, n ($n \leq 10^5$), representing the number of food items on the table. The second line contains n space-separated positive integers. The i^{th} integer, t_i ($-1000 \leq t_i \leq 1000$), on this line represents the tastiness of the i^{th} food item from the left on the table.

Output

For each test case output the maximum sum of tastiness Nom Chomsky can enjoy within four bites or less.

Samples

Input	Output
1 -1	0
2 1 -2	1
11 1 -9 5 -1 2 -3 1 -4 3 -4 1	11
9 2 -3 2 -3 3 -1 5 -5 3	14
9 10 -5 9 -100 30 -10 9 -6 10	63

Solution for the fourth sample test case with bites bolded: **2** -3 **2** -3 **3 -1 5** -5 **3**
Solution for the fifth sample test case with bites bolded: **10 -5 9** -100 **30** -10 **9** -6 **10**

Problem C: Garlic Bread Memes

Filename: `garlicbread`
Time limit: `3 seconds`

While browsing a meme website, you come across the following post:

"You have been visited by the Great Garlic Bread! Upvote this post and some of the comments on this post in 3 seconds per test and luck shall rain down on you! But, you must upvote in a way that makes garlic bread look like it has a *long and increasingly positive* history. The comments are indexed from 1 to ***N*** in order of the time each comment was posted. Comment ***i*** has some garlic bread positivity rating ***p(i)***, representing how much positive garlic bread emotions are expressed in comment ***i***. Upvote this post and some *maximum-sized* subset of the comments ***S*** (sorted by indices) such that the slope between chosen comment S_i and S_{i+1}, defined as $\frac{p(S_{i+1})-p(S_i)}{S_{i+1}-S_i}$, is not less than ***K*** for the luck to rain down on you."

This post is likely just spam. In case you DO decide to upvote the post and the comments with the restrictions listed, what is the most number of comments you can upvote?

Input

The first line of input consists of two space separated integers, ***n*** ($1 \le n \le 1000$), the number of comments and ***k*** ($-10^9 \le k \le 10^9$), the minimum slope the subset of comments is allowed to have between adjacent comments. Line 2 describes the function ***p***, where the *i*th integer equals ***p(i)*** ($-10^9 \le p(i) \le 10^9$). All integers on both lines will be separated by spaces.

Output

Output the maximum number of comments you would upvote while following the restrictions of the post.

Samples

Input	Output
10 0 2 12 4 6 5 3 10 -5 8 10	5
10 -1 2 12 4 6 5 3 10 -5 8 10	7

Sample Input 1 Explanation: Upvote comments 1, 3, 4, 7, and 10. The slopes between consecutive comments in this sequence are 1, 2, 4/3, and 0, respectively, all greater than or equal to 0.

Sample Input 2 Explanation: Upvote comments 1, 3, 4, 5, 7, 9, and 10. The slopes between consecutive comments in this sequence are 1, 2, -1, 5/2, -1, and 2, respectively, all greater than or equal to -1.

Problem D: Universal Locker Rental 2

Filename: `locker2`
Time limit: `1 second`

Universal Studios lets guests put their belongings in lockers while they enjoy thrill rides. As a courtesy, the lockers are free for up to one hour. After that, guests get charged a rather large amount of money for using a locker. Normally, you would put your belongings in a locker, enjoy a single ride and then collect your belongings. However, the process of checking out a locker and retrieving your items is rather laborious. Thus, it would be nice to minimize the number of times you have to check a locker. If the ride times are low enough, you could ride maybe two or three rides before having to retrieve your belongings and still not get charged at all.

Write a program that, given a list of the ride times (wait plus ride time) for all the rides you want to enjoy for a day, determines the minimum number of times you will need to check out a locker without paying any money. ***You may select any order to ride the rides that minimizes this number.*** For the purposes of this problem, assume that it takes 7 ½ minutes to travel from a locker to any ride and 7 ½ minutes to travel from any ride back to a locker.

Input

The first line of input contains a single positive integer, n ($1 \le n \le 40$), representing the number of rides you'd like to enjoy for the day. This is followed by n lines of input, each of which has a single positive integer, t ($15 \le t \le 45$), representing one of the ride times.

Output

On a single line, output the minimum number of times you will have to check out a locker to avoid paying for the locker at all.

Samples

Input	Output
5 15 20 35 30 20	3
4 15 20 25 35	3

For the first sample, the ordering (15, 30), (20, 20), (35), shows that three locker check outs suffice.

Problem E: Message Spam

Filename: `spam`
Time limit: `1 second`

Somehow people keep on lending their phones to Zachary. The most logical thing to do in this situation is to go to the GroupMe CPSI 2017 unofficial chat from their phone and start spamming messages. When they get their phone back and see how they got roasted by the other campers, they will conclude that redemption is impossible and leave the group chat of their own volition.

Gabriel wants to add them back to group chat, but he needs know whose phones were in Zach's possession. He suspects that anyone who has submitted three or more identical messages is a victim. The messages need not necessarily be adjacent. For the purposes of this problem, when comparing two messages, do a ***case sensitive*** comparison. Thus, "Hi" and "hi" should NOT be treated as identical messages.

Input

The input begins with a single positive integer ***m*** (***m*** ≤ 100) the number of messages. Following this on each line will be a message of the form ***<name>***: ***<text>***. ***<name>*** will contain only alphabetic characters while ***<text>*** can contain alphabetic characters, numbers, spaces, and the symbols **! @ ? " , ' . ( )**. ***<name>*** will have no more than 20 characters while ***<text>*** will have no more than 100. The first letter of ***<name>*** will always be uppercase. There will never be two or more spaces in a row or trailing spaces. ***<name>*** is always followed by a colon and a space (:).

Output

On separate lines, output the names of people in the chat who submitted the same message at least 3 times in alphabetical order by name. If there are no such people, output -1.

Samples

Input	Output
11 C: hey B: what's up A: the sky B: what's up A: the sky B: what's up A: the sky C: hey A: ho A: ho A: ho	A B

Input
3 Miles: anyone want to kill a giant roach?? lol please Ian: Call 911 Kurt: Evacuate the building
Output
-1

Input
22 Ethan: Hec 101 like normal? Right Ethan: ? Jacob: ? Chris: Im da map Chris: Im da map Chris: Im da map Chris: Im da map Chris: Im da map Kurt: ? Chris: Im da map Chris: Im da map Chris: Im da map Chris: Im da map Jeffrey: wait are you da map? Jeffrey: im not sure Jeffrey: you haven’t made it clear enough Allen: burn the map smh Kian: Be proud of yourself chris Kian: You are the map Lina: I’m pretty sure "map" is an acronym for "most annoying person" Dylan: Rip Chris William: @Chris Pan might as well just leave the camp now
Output
Chris

Problem F: Superprime

Filename: `superprime`
Time limit: `1 second`

Arup loves prime numbers. Not only that, but he loves prime numbers IN prime numbers. In a quest to get his name in the number theory books, he's come up with his own definition: a super prime number.

A super prime number is one such that each "prefix" of the number is prime as well. Consider the prime number 7393. Since each of its 4 prefixes, 7, 73, 739 and 7393 are prime, it is a super prime number!

Arup would like to enlist your help in studying super primes so that others may find them interesting and study them as well so that eventually he'll make a contribution to number theory books.

In particular, he'd like for you to write a program that takes in an integer ***k*** and returns the ***k***th super prime number, in numerical order.

Input

The input contains a single positive integer, ***k***. The input will be such that it is guaranteed that the ***k***th super prime number exists.

Output

On a single line, output the ***k***th super prime number, in numerical order.

Samples

Input	Output
7	31
23	599

Note: The first seven super primes are 2, 3, 5, 7, 23, 29 and 31.

Problem G: Clear the Board

Filename: `board`
Time limit: `2 seconds`

Matthew is playing a game called Clear the Board. In this game, Matthew is given an 8x8 board with pieces on it. Matthew's objective is to clear the board in as few moves as possible, where a move consists of removing every piece on a single diagonal.

Input

The first line of input is a single integer, ***n*** (1 ≤ ***n*** ≤ 64), that denotes how many pieces are on the board. The next ***n*** lines consist of two integers, ***x*** and ***y*** (1 ≤ ***x***, ***y*** ≤ 8), that describe the ***x*** and ***y*** coordinates of that piece on the board.

Output

On a line by itself, output the minimum number of moves required for Matthew to clear the board.

Samples

Input	Output
3 1 1 3 3 3 5	2

Input	Output
4 2 2 4 4 2 5 5 2	2

Explanation of Sample Input 1:
Matthew uses one move to clear the pieces at (1, 1) and (3, 3). He uses another a move to clear the piece at (3, 5).
Explanation of Sample Input 1:
Matthew uses one move to clear the pieces at (2, 2) and (4, 4). He uses another a move to clear the pieces at (2, 5) and (5, 2).

Problem H: River Delta Tour

Filename: `river`
Time limit: `3 seconds`

In Libraland river deltas resemble complete binary trees. In other words, each delta will have 2^n-1 intersections and 2^n-2 joining streams for some positive integer n. However, there actually are no oceans or lakes for the deltas to drain into. Instead, river deltas just flow into some other river delta that branches the same number of times. A delta with 2^n-1 intersections will share 2^{n-1} of them with the other delta. These deltas are famous for their beauty and tours of them are quite popular, but some streams are more beautiful than others. Furthermore, no one wants to visit the same stream more than once in tour (but visiting the same intersection is fine). With this in mind, LibraTours LLC wants to know the maximum beauty possible for a tour starting from the entrance of a river's delta.

Input

Each test case starts with line containing a single integer n ($2 \leq n \leq 15$). Proceeding this will be $2n$ -2 lines which describe the beauty values b_i ($1 \leq b_i \leq 500$) of all the streams. The first n-1 lines describe the first river delta. Each line corresponds to a depth in the binary tree representation of the river delta. The first line of the description will have 2 beauty values, the beauty values of the streams directly splitting off from the river entrance. The number of beauty values per line doubles until the line containing 2^{n-1} values. The next n-1 lines describe the second river delta in a similar fashion, but upside down (refer to diagrams of samples).

Output

Output a single integer, the maximum beauty tour possible starting from the entrance of the first river delta that does not revisit any streams.

Samples

Input	Output
3 1 6 1 1 7 5 1 2 6 5 2 1	30
4 1 30 1 1 30 1 4 5 5 4 30 30 1 2 6 4 4 4 30 30 4 3 1 2 3 1 2 2	180
2 1 1 1 1	4

Diagram for sample 1:

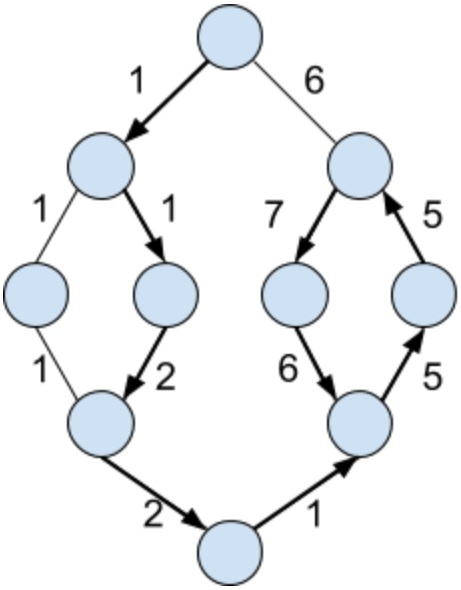
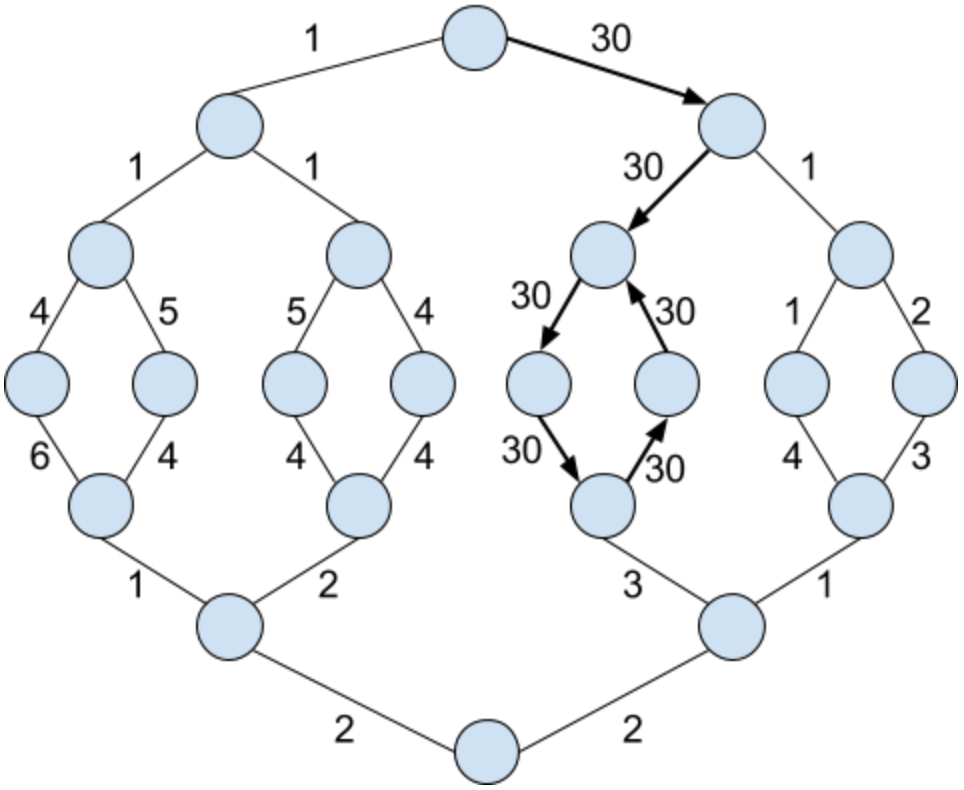


Diagram for sample 2:



Problem I: Magnetic Strip

Filename: `strip`
Time limit: `5 seconds`

While playing with the magnetic strips that came with his mini whiteboard, Gabe realized that the strips were made from several equal length bands of positive or negative magnets with no particular order. The strips are only magnetic on one side and Gabe enjoys folding them and trying to hold them together by matching positive bands with negative bands. In order to make the strongest bond he would like to know the largest contiguous section on a strip that can be perfectly paired with another section on the same strip. Folding can be done both in between bands and on the center of a band.

Input

Each test case consists of a single string of '+' and '-' characters denoting positive and negative bands on the magnetic strip. The length of this string is at least 1 and no more than 10^5.

Output

On a line by itself, output the maximum length section that can be paired with another section on the same strip.

Samples

Input	Output
`++`	`0`
`++-+-+-`	`3`
`+--+++-----++++---+++`	`8`
`-+--+`	`2`

Solution for third sample test case: