Problem A: Maria's Midpoint Mystery

Filename: `midpoint`
Timelimit: `2 seconds`

One day while bored, Maria decided to take a piece of paper and place ***N***+1 dots in what she thinks were random integer positions along a straight line on the paper. Feeling a little fiddly, she also marked the midpoints between each adjacent dot on the line.

The next day she woke up to find that the dots have been erased. She wants to restore the dots on the paper at integer coordinates, but has no clue where to actually put the dots. Really any configuration of dots will do, so long as 1) the midpoints of the previous paper are also midpoints of the new dots, 2) no dot shares a coordinate, and 3) all the new dots lie on integer coordinates. But, as Maria knows someone messed with her paper, she also knows some of the midpoints might have also been tampered with, so she wonders if it's even possible to reconstruct the dots on integer coordinates.

Input

The first line of input contains a single positive integer ***n*** ($2 \le n \le 10^5$), representing the number of midpoints. The next line contains ***n*** space separated integers; the *i*th integer on this line denotes the signed distance the midpoint between point i and point i+1 is from the center of the paper. These integers will be unique, ordered from least to greatest, and within in the range [-10^{18}, 10^{18}], with integers less than 0 representing points to the left of the origin of the line and integers greater than 0 representing points to the right of the origin of the line.

Output

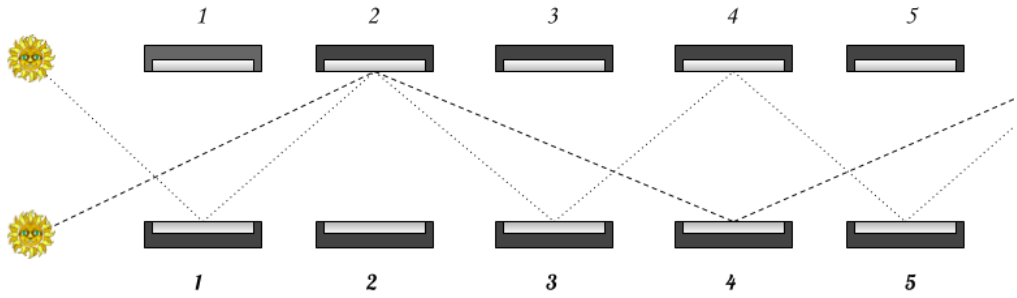
If it is impossible to draw dots on the lines at integer coordinates such that the dots match up with the paper, output `-1`. Otherwise, output two integers, representing the leftmost and rightmost integer positions where the first dot might be (it's fairly easy to tell the positions of the rest of the dots from this one coordinate).

Samples

Input	Output
5 2 6 14 20 25	1 1
4 -3 3 7 13	-8 -6
7 -7 -2 2 8 12 20 27	-1

Problem B: Hall of Mirrors

Filename: `mirror`
Timelimit: `1 second`



While working their way through the Earth Temple, Link and Medli came across a room filled with two rows of mirrors. The room has two rays of sun shining down from the ceiling. Link can use his mirror shield and Medli her golden harp to direct light to the mirrors. Some mirrors have glyphs on them that give off a blue glow while light is reflected off them. Link suspects all glyphs need to be glowing to solve the puzzle of room, so they want to know the minimum number of rays necessary to light up all the glyphs on the mirrors. There are ***n*** mirrors in both the top and bottom rows each row is separately numbered from 1 to ***n***. The center of the ith mirror on the top row is at location (i, 1) while the center of the ith mirror on the bottom row is at (i,0). The two sunspots are at locations (0,1) and (0, 0). Multiple rays of light can be reflected from the same sunspot. Rays can light up glyphs regardless of the number of times it has been reflected off a mirror. Since Link and Medli use Z-targeting, they can only aim for the centers of mirrors.

Input

The first line of input will contain three space separated integers, n ($1 \le n \le 10^5$) the number mirrors per row. The next line will first contain an integer t ($0 \le t \le n$), which is followed t integers t_i ($1 \le t_i \le n$), the mirrors on the top row that hold glyphs. The next line contains an integer b ($0 \le b \le n$) followed by b integers b_i ($1 \le b_i \le n$) the mirrors of the bottom row that hold glyphs. Glyphs will be given in increasing order.

Output

On a line by itself output a single integer, the minimum number of rays that need to be reflected to light up all the glyphs.

Samples

Input	Output
5 1 4 4 1 3 4 5	2
5 2 3 5 0	1
30 0 3 8 16 24	1
1 0 0	0

Problem C: N Queens

Filename: `nqueens`
Time limit: `2 seconds`

We wish to solve the N Queens problem - how to place n queens on an n by n chessboard so that no two are attacking each other. However, we wish to only find the lexicographical first solution. Let $\{p_1, p_2, p_3, \ldots, p_N\}$ be a permutation of the set {1, 2, 3, …, N}. We let each permutation describe a placement of queens as follows: p_i represents the column in which the queen on row i is placed. Thus, the permutation {2, 4, 1, 3} represents queens in row 1 column 2, row 2 column 4, row 3 column 1 and row 4 column 3, as shown below:

	Q		
			Q
Q			
		Q	

This is the first lexicographical solution for N = 4, since none of the queens shown above share the same row, column or diagonal. (Note: To compare two solutions in lexicographical ordering, find the first corresponding number that differs. The one that comes first is the one that has a lower number for the first differing number. Thus, 2, 4, 1, 3 comes before 3, 1, 2, 4, but after 2, 3, 4, 1.)

Input

The input consists of a single positive integer, ***n*** ($4 \leq n \leq 15$), representing that the chess board for the input case is size ***n*** x ***n***.

Output

Output on a single line ***n*** space separated integers representing the first lexicographical solution to the ***n*** queens problem as described above.

Samples

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

Problem D: Presidential Security

Filename: `security`
Time limit: `1 second`

After hearing that you had attended SI@UCF Programming Competition Camp, the White House was so impressed, that they hired you to help lay out the communication network when the President traveled. The president only likes staying in hotels that have all of their rooms on each floor on a row. The rooms are numbered left to right, and no floor has more than 100 rooms, so the last two digits of each room uniquely identify where the room is within a given floor. All digits beyond the last two digits signify the floor of the room. For example, room 6287 is the 88th room from the left (starting from 0) on the 62nd floor. Unfortunately, the president doesn't travel light. His party may end up taking several rooms at a hotel, and often times, to avoid suspicion, the team stays scattered throughout the hotel.

For example, consider a hotel with 4 floors and 8 rooms per floor (shown below) and the President's party staying in rooms 101, 203, 207, 402 and 406. The President would like you to set up wired communication that links all five rooms, either directly or indirectly, in the shortest time possible.

400	**401**	**402**	**403**	**404**	**405**	**406**	**407**
300	**301**	**302**	**303**	**304**	**305**	**306**	**307**
200	**201**	**202**	**203**	**204**	**205**	**206**	**207**
100	**101**	**102**	**103**	**104**	**105**	**106**	**107**

For all hotels with this layout, you've discovered that you can wire any two rooms directly and that the amount of time (in minutes) it will take you to lay a single wire between two rooms is ***a(dx) + b(dy)***, where ***a*** and ***b*** are given constants (different for each hotel) and ***dx*** represents how many rooms separate the two rooms on the x axis, which runs horizontal to the ground, and ***dy*** represents how many rooms separate the two rooms on the y axis, which runs vertically. For example, if ***a*** = 2 and ***b*** = 9, then we could connect the following sets of rooms for the hotel above:

101 and 203, time = 2(3 - 1) + 9(2 - 1) = 13
203 and 402, time = 2(3 - 2) + 9(4 - 2) = 20
203 and 207, time = 2(7 - 3) + 9(2 - 2) = 8
402 and 406, time = 2(6 - 2) + 9(4 - 4) = 8

in 13 + 20 + 8 + 8 = 49 minutes. This arrangement allows all pairs of room to communicate and all other alternate arrangements would take you equal or more time to set up as this one.

Write a program to calculate the minimum time it'll take you to set up a network the president desires for various hotels.

Input

The first line of input will contain three space separated positive integers, ***n***, (***n*** ≤ 100), representing the number of rooms the President's party is taking in that hotel, ***a***, (***a*** ≤ 100) and ***b*** (***b*** ≤ 100), representing the two given constants for the hotel in the input case.

The next ***n*** lines of the input contain one positive integer, each, representing one of the room numbers in the President's party. Each of these integers is guaranteed to be distinct and in between 100 and 9999, inclusive.

Output

On a line by itself, output the minimum number of minutes it will take you to set up the network for the president.

Samples

Input	Output
5 2 9 101 203 207 402 406	49
3 1 100 100 199 200	199

Problem E: Blaze Pizza Wait Times

Filename: `blaze`
Time limit: `1 second`

Friday's dinner at Blaze Pizza was incredibly tasty. Unfortunately, it was so tasty that everyone else also goes to Blaze Pizza, which means that you have to wait a while. It would be nice to know just how long you'll have to wait in line so you can figure out whether or not you have time to play an entire level of your new video game before you have to give your order.

For the purposes of this problem, your wait time is based upon the pizza orders of all of the people in front of you in line. The crust, sauce and cheese for any pizza take 15 seconds to prepare. Each topping added to the pizza takes 2 seconds to prepare. Each order is handled sequentially, thus, your wait time is simply the sum of the order times of all of the people in line in front of you. Write a program to determine your wait time in hours, minutes and seconds.

Input

The first line of input contains a single positive integer, n ($1 \le n \le 100$), representing the number of people in line in front of you. This is followed by n lines of input, where the i^{th} of these lines has a single positive integer, t_i ($1 \le t_i \le 1000$), representing the number of toppings on the pizza ordered by the i^{th} person in line.

Output

Output on a single line three space separated values: the number of hours, minutes and seconds you'll have to wait to order your pizza. The number of minutes and seconds printed must be in between 0 and 59, inclusive.

Samples

Input	Output
3 4 5 2	0 1 7
5 1000 1000 1000 1000 1000	2 47 55

Problem F: Blaze Pizza Combinations

Filename: `combos`
Time limit: `1 second`

Blaze Pizza allows you to choose and combination of toppings from their list. The company policy is that you must order your toppings in alphabetical order. Thus, each possible order can be listed in lexicographical order. Namely, when comparing two orders, we compare the first topping in each order, then the second, etc. until we get to our first pair that isn't the same. The order that comes first lexicographically is the order that has the first alphabetic topping in the pair that doesn't match. If an order X is a proper subset of an order Y, then order X comes first alphabetically. For example, if the toppings are {mushrooms, pepperoni, spinach}, the seven possible non-empty combinations of toppings in lexicographical order are:

{mushrooms}, {mushrooms, pepperoni}, {mushrooms, pepperoni, spinach},
{mushrooms, spinach}, {pepperoni}, {pepperoni, spinach}, {spinach}

In ordering pizzas, the SI@UCF campers have decided that it would be more efficient if people just gave their order with a single positive integer: the lexicographical rank of the toppings they would like on their pizza. Unfortunately, the Blaze Pizza staff are having trouble converting these ranks to a set of toppings. Write a program to help the Blaze Pizza staff decipher this strange language the SI@UCF campers have devised.

Input

The first line of input contains a single positive integer, n ($1 \leq n \leq 20$), representing the number of toppings Blaze Pizza has available. The second line of input contains n space separated strings representing the toppings available. Each of these will contain in between 1 and 20 lowercase letters, inclusive. The third line of input has a single positive integer, r ($1 \leq r \leq 2^n - 1$), representing the numerical order (lexicographical rank amongst all possible non-empty order of pizza toppings) given by the SI@UCF camper.

Output

Output the set of toppings that correspond to the input rank, based on the list of toppings available. Each topping on your list should be separated by a single space and listed in alphabetical order.

Samples

Input	Output
3 mushrooms pepperoni spinach 6	pepperoni spinach
5 spinach pepperoni mushrooms sausage bananapeppers 17	mushrooms

Problem G: Universal Locker Rental

Filename: `locker`
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, in the order that you want to ride them, determines the minimum number of times you will need to check out a locker without paying any money. 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	4
4 15 20 25 35	3

Problem H: Top 25

Filename: `top`
Timelimit: `5 seconds`

In College Football, many different sources create a list of the Top 25 (or, Top ***n***) teams in the country. Since it's subjective, these lists often differ, but they're usually very similar. Your job is to compare two of these lists, and determine where they are similar. In particular, you are to partition them into sets, where each set represents the same contiguous positions in both lists, and has the same teams, and is as small as possible. If the lists agree completely, you'll have n sets, where n is the number of teams in each list. For example consider these two lists:

For example consider these two lists:

A	A
B	C
C	D
D	B
E	E

In this case, there are 3 sets: A, BCD, and E.

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. Each test case will begin with an integer ***n*** ($1 \le n \le 1{,}000{,}000$), indicating the number of teams ranked. The next ***n*** lines will hold the first list, in order. The team names will appear one per line, and consist of between 1 and 8 capital letters only. After this will be ***n*** lines, in the same format, indicating the second list. Both lists will contain the same team names, and all ***n*** team names will be unique. .

Output

Output the size of each set, in order, one per line. Do not output any spaces, and do not output blank lines between numbers.

Samples

Input	Output
5 A B C D E A C D B E	1 3 1
3 RED BLUE ORANGE RED BLUE ORANGE	1 1 1
3 MOE LARRY CURLY CURLY MOE LARRY	3