Problem A. Digits Are Not Just Characters

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Mr. Manuel Majorana Minore made a number of files with numbers in their names. He wants to have a list of the files, but the file listing command commonly used lists them in an order different from what he prefers, interpreting digit sequences in them as ASCII code sequences, not as numbers. For example, the files `file10`, `file20` and `file3` are listed in this order.

Write a program which decides the orders of file names interpreting digit sequences as numeric values.

Each file name consists of uppercase letters (from '`A`' to '`Z`), lowercase letters (from '`a`' to '`z`'), and digits (from '`0`' to '`9`').

A file name is looked upon as a sequence of items, each being either a letter or a number. Each single uppercase or lowercase letter forms a letter item. Each consecutive sequence of digits forms a number item.

Two item are ordered as follows.

- Number items come before letter items.
- Two letter items are ordered by their ASCII codes.
- Two number items are ordered by their values when interpreted as decimal numbers.

Two file names are compared item by item, starting from the top, and the order of the first different corresponding items decides the order of the file names. If one of them, say A, has more items than the other, B, and all the items of B are the same as the corresponding items of A, B should come before.

For example, three file names in Sample Input 1, `file10`, `file20`, and `file3` all start with the same sequence of four letter items '`f`', '`i`', '`l`', and '`e`', followed by a number item, 10, 20, and 3, respectively. Comparing numeric values of these number items, they are ordered as `file3` $<$ `file10` $<$ `file20`.

Input

The integer n in the first line of the input gives the number of file names (s_1 through s_n) to be compared with the file name given in the next line (s_0). Here, n satisfies $1 \le n \le 1000$. The following $n+1$ lines are file names, s_0 through s_n, one in each line. They have at least one and no more than nine characters. Each of the characters is either an uppercase letter, a lowercase letter, or a digit.

Sequences of digits in the file names never start with a digit zero (0).

Output

For each of the file names, s_1 through s_n, output one line with a character indicating whether it should come before s_0 or not. The character should be '`-`' if it is to be listed before s_0; otherwise, it should be '`+`', including cases where two names are identical.

Examples

standard input	standard output
2	+
file10	-
file20	
file3	

standard input	standard output
11	-
X52Y	-
X	-
X5	+
X52	+
X52Y	-
X52Y6	-
32	+
ABC	+
XYZ	-
x51y	+
X8Y	
X222	

Problem B. Arithmetic Progressions

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	512 mebibytes

An arithmetic progression is a sequence of numbers $a_1, a_2, \ldots, a_k$ where the difference of consecutive members $a_{i+1} - a_i$ is a constant ($1 \le i \le k-1$). For example, the sequence 5, 8, 11, 14, 17 is an arithmetic progression of length 5 with the common difference 3.

In this problem, you are requested to find the longest arithmetic progression which can be formed selecting some numbers from a given set of numbers. For example, if the given set of numbers is $\{0, 1, 3, 5, 6, 9\}$, you can form arithmetic progressions such as 0, 3, 6, 9 with the common difference 3, or 9, 5, 1 with the common difference -4. In this case, the progressions 0, 3, 6, 9 and 9, 6, 3, 0 are the longest.

Input

The input consists of a single test case of the following format.

n

$v_1\ v_2\ \ldots v_n$

n is the number of elements of the set, which is an integer satisfying $2 \le n \le 5000$. Each v_i ($1 \le i \le n$) is an element of the set, which is an integer satisfying $0 \le v_i \le 10^9$. v_i's are all different, i.e., $v_i = v_j$ if and only if $i = j$.

Output

Output the length of the longest arithmetic progressions which can be formed selecting some numbers from the given set of numbers.

Examples

standard input	standard output
6 0 1 3 5 6 9	4
7 1 4 7 3 2 6 5	7
5 1 2 4 8 16	2

Problem C. Emergency Evacuation

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

The Japanese government plans to increase the number of inbound tourists to forty million in the year 2020, and sixty million in 2030. Not only increasing touristic appeal but also developing tourism infrastructure further is indispensable to accomplish such numbers.

One possible enhancement on transport is providing cars extremely long and/or wide, carrying many passengers at a time. Too large a car, however, may require too long to evacuate all passengers in an emergency. You are requested to help estimating the time required. The car is assumed to have the following seat arrangement.

- A center aisle goes straight through the car, directly connecting to the emergency exit door at the rear center of the car.
- The rows of the same number of passenger seats are on both sides of the aisle. A rough estimation requested is based on a simple step-wise model. All passengers are initially on a distinct seat, and they can make one of the following moves in each step.
- Passengers on a seat can move to an adjacent seat toward the aisle. Passengers on a seat adjacent to the aisle can move sideways directly to the aisle.
- Passengers on the aisle can move backward by one row of seats. If the passenger is in front of the emergency exit, that is, by the rear-most seat rows, he/she can get off the car.

The seat or the aisle position to move to must be empty; either no other passenger is there before the step, or the passenger there empties the seat by moving to another position in the same step. When two or more passengers satisfy the condition for the same position, only one of them can move, keeping the others wait in their original positions.

The leftmost figure of Figure C.1 depicts the seat arrangement of a small car given in Sample Input 1. The car have five rows of seats, two seats each on both sides of the aisle, totaling twenty. The initial positions of seven passengers on board are also shown.

The two other figures of Figure C.1 show possible positions of passengers after the first and the second steps. Passenger movements are indicated by fat arrows. Note that, two of the passengers in the front seat had to wait for a vacancy in the first step, and one in the second row had to wait in the next step.

Your task is to write a program that gives the smallest possible number of steps for all the passengers to get off the car, given the seat arrangement and passengers initial positions.

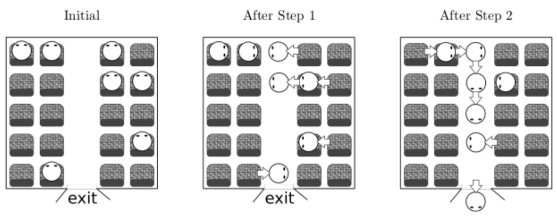

Figure C.1. Simple Model

Input

The input consists of a single test case of the following format.

$r\ s\ p$

$i_1\ j_1$

$\ldots$

$i_p\ j_p$

Here, r is the number of passenger seat rows, s is the number of seats on each side of the aisle, and p is the number of passengers. They are integers satisfying $1 \le r \le 500$, $1 \le s \le 500$, and $1 \le p \le 2rs$.

The following p lines give initial seat positions of the passengers. The k-th line with i_k and j_k means that the k-th passenger's seat is in the i_k-th seat row and it is the j_k-th seat on that row. Here, rows and seats are counted from front to rear and left to right, both starting from one. They satisfy $1 \le i_k \le r$ and $1 \le j_k \le 2s$. Passengers are on distinct seats, that is, $i_k \ne i_l$ or $j_k \ne j_l$ holds if $k \ne l$.

Output

The output should be one line containing a single integer, the minimum number of steps required for all the passengers to get off the car.

Examples

standard input	standard output

5 2 7
1 1
1 2
1 3
2 3
2 4
4 4
5 2
	9

500 500 16
1 1
1 2
1 999
1 1000
2 1
2 2
2 999
2 1000
3 1
3 2
3 999
3 1000
499 500
499 501
499 999
499 1000
	1008

Problem D. Shortest Common Non-Subsequence

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

A subsequence of a sequence P is a sequence that can be derived from the original sequence P by picking up some or no elements of P preserving the order. For example, "`ICPC`" is a subsequence of "`MICROPROCESSOR`".

A common subsequence of two sequences is a subsequence of both sequences. The famous longest common subsequence problem is finding the longest of common subsequences of two given sequences.

In this problem, conversely, we consider the shortest common non-subsequence problem: Given two sequences consisting of 0 and 1, your task is to find the shortest sequence also consisting of 0 and 1 that is a subsequence of neither of the two sequences.

Input

The input consists of a single test case with two lines. Both lines are sequences consisting only of 0 and 1. Their lengths are between 1 and 4000, inclusive.

Output

Output in one line the shortest common non-subsequence of two given sequences. If there are two or more such sequences, you should output the lexicographically smallest one. Here, a sequence P is lexicographically smaller than another sequence Q of the same length if there exists k such that $P_1 = Q_1, \ldots, P_{k-1} = Q_{k-1}$, and $P_k < Q_k$, where S_i is the i-th character of a sequence S.

Example

standard input	standard output
0101 1100001	0010
101010101 010101010	000000
11111111 00000000	01

Problem E. Eulerian Flight Tour

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

You have an airline route map of a certain region. All the airports in the region and all the non-stop routes between them are on the map. Here, a non-stop route is a flight route that provides non-stop flights in both ways.

Named after the great mathematician Leonhard Euler, an Eulerian tour is an itinerary visiting all the airports in the region taking a single flight of every non-stop route available in the region.

To be precise, it is a list of airports, satisfying all of the following.

- The list begins and ends with the same airport.
- There are non-stop routes between pairs of airports adjacent in the list.
- All the airports in the region appear at least once in the list. Note that it is allowed to have some airports appearing multiple times.
- For all the airport pairs with non-stop routes in between, there should be one and only one adjacent appearance of two airports of the pair in the list in either order.

It may not always be possible to find an Eulerian tour only with the non-stop routes listed in the map. Adding more routes, however, may enable Eulerian tours. Your task is to find a set of additional routes that enables Eulerian tours.

Input

The input consists of a single test case.

n m

a_1 b_1

...

a_m b_m

n ($3 \le n \le 100$) is the number of airports. The airports are numbered from 1 to n. m ($0 \le m \le n(n-1)$) is the number of pairs of airports that have non-stop routes. Among the m lines following it, integers a_i and b_i on the i-th line of them ($1 \le i \le m$) are airport numbers between which a non-stop route is operated. You can assume $1 \le a_i < b_i \le n$, and for any $i \ne j$, either $a_i \ne a_j$ or $b_i \ne b_j$ holds.

Output

Output a set of additional non-stop routes that enables Eulerian tours. If two or more different sets will do, any one of them is acceptable. The output should be in the following format.

k

c_1 d_1

...

c_k d_k

k is the number of non-stop routes to add, possibly zero. Each of the following k lines should have a pair of integers, separated by a space. Integers c_i and d_i in the i-th line ($c_i < d_i$) are airport numbers specifying that a non-stop route is to be added between them. These pairs, (c_i, d_i) for $1 \le i \le k$, should be distinct and should not appear in the input. If adding new non-stop routes can never enable Eulerian tours, output -1 in a line.

Examples

standard input	standard output
4 2	2
1 2	1 4
3 4	2 3
6 9	-1
1 4	
1 5	
1 6	
2 4	
2 5	
2 6	
3 4	
3 5	
3 6	
6 7	3
1 2	1 5
1 3	2 4
1 4	2 5
2 3	
4 5	
4 6	
5 6	
4 3	-1
2 3	
2 4	
3 4	
5 5	0
1 3	
1 4	
2 4	
2 5	
3 5	

Problem F. Fair Chocolate-Cutting

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

You are given a flat piece of chocolate of convex polygon shape. You are to cut it into two pieces of precisely the same amount with a straight knife.

Write a program that computes, for a given convex polygon, the maximum and minimum lengths of the line segments that divide the polygon into two equal areas.

The figures below correspond to first two sample inputs. Two dashed lines in each of them correspond to the equal-area cuts of minimum and maximum lengths.

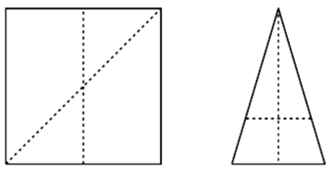
Figure F.1. Sample Chocolate Pieces and Cut Lines

Input

The input consists of a single test case of the following format.

n

x_1 y_1

...

x_n y_n

The first line has an integer n, which is the number of vertices of the given polygon. Here, n is between 3 and 5000, inclusive. Each of the following n lines has two integers x_i and y_i, which give the coordinates (x_i, y_i) of the i-th vertex of the polygon, in counterclockwise order. Both x_i and y_i are between 0 and 10^5, inclusive.

The polygon is guaranteed to be simple and convex. In other words, no two edges of the polygon intersect each other and interior angles at all of its vertices are less than 180 degrees.

Output

Two lines should be output. The first line should have the minimum length of a straight line segment that partitions the polygon into two parts of the equal area. The second line should have the maximum length of such a line segment. The answer will be considered as correct if the values output have an absolute or relative error less than 10^{-6}.

Examples

standard input	standard output

4
0 0
10 0
10 10
0 10
	10
	14.142135623730950488

3
0 0
6 0
3 10
	4.2426406871192851464
	10.0

5
0 0
99999 20000
100000 70000
33344 63344
1 50000
	54475.580091580027976
	120182.57592539864775

6
100 350
101 349
6400 3440
6400 3441
1200 7250
1199 7249
	4559.2050019027964982
	6216.7174287968524227

Problem G. What Goes Up Must Come Down

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Several cards with numbers printed on them are lined up on the table.

We'd like to change their order so that first some are in non-decreasing order of the numbers on them, and the rest are in non-increasing order. For example, (1, 2, 3, 2, 1), (1, 1, 3, 4, 5, 9, 2), and (5, 3, 1) are acceptable orders, but (8, 7, 9) and (5, 3, 5, 3) are not.

To put it formally, with n, the number of cards and bi, the number printed on the card, at the i-th position ($1 \le i \le n$) after reordering, there should exist $k \in \{1, \ldots, n\}$ such that ($b_i \le b_{i+1}$ for all $i \in \{1, \ldots, k-1\}$) and ($b_i \ge b_{i+1}$ for any $i \in \{k, \ldots, n-1\}$) hold. For reordering, the only operation allowed at a time is to swap the positions of an adjacent card pair. We want to know the minimum number of swaps required to complete the reorder.

Input

The input consists of a single test case of the following format.

n

$a_1 \ldots a_n$

An integer n in the first line is the number of cards ($1 \le n \le 10^5$). Integers a_1 through a_n in the second line are the numbers printed on the cards, in the order of their original positions ($1 \le a_i \le 10^5$).

Output

Output in a line the minimum number of swaps required to reorder the cards as specified.

Examples

standard input	standard output
7 3 1 4 1 5 9 2	3
9 10 4 6 3 15 9 1 1 12	8
8 9 9 8 8 7 7 6 6	0
6 8 7 2 5 4 6	4

Problem H. Four-Coloring

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

You are given a planar embedding of a connected graph. Each vertex of the graph corresponds to a distinct point with integer coordinates. Each edge between two vertices corresponds to a straight line segment connecting the two points corresponding to the vertices. As the given embedding is planar, the line segments corresponding to edges do not share any points other than their common endpoints. The given embedding is organized so that inclinations of all the line segments are multiples of 45 degrees. In other words, for two points with coordinates (x_u, y_u) and (x_v, y_v) corresponding to vertices u and v with an edge between them, one of $x_u = x_v$, $y_u = y_v$, or $|x_u - x_v| = |y_u - y_v|$ holds.

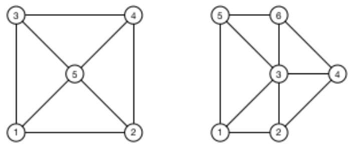

Figure H.1. Sample Input 1 and 2

Your task is to color each vertex in one of the four colors, $\{1,\ 2,\ 3,\ 4\}$, so that no two vertices connected by an edge are of the same color. According to the famous four color theorem, such a coloring is always possible. Please find one.

Input

The input consists of a single test case of the following format.

n m

x_1 y_1

...

x_n y_n

u_1 v_1

...

u_m v_m

The first line contains two integers, n and m. n is the number of vertices and m is the number of edges satisfying $3 \leq n \leq m \leq 10^4$. The vertices are numbered 1 through n. Each of the next n lines contains two integers. Integers on the v-th line, x_v $(0 \leq x_v \leq 1000)$ and y_v $(0 \leq y_v \leq 1000)$, denote the coordinates of the point corresponding to the vertex v. Vertices correspond to distinct points, i.e., $(x_u, y_u) \neq (x_v, y_v)$ holds for $u = v$. Each of the next m lines contains two integers. Integers on the i-th line, u_i and v_i , with $1 \leq u_i < v_i \leq n$, mean that there is an edge connecting two vertices u_i and v_i.

Output

The output should consist of n lines. The v-th line of the output should contain one integer $c_v \in \{1, 2, 3, 4\}$ which means that the vertex v is to be colored c_v. The output must satisfy $c_u \neq c_v$ for every edge connecting u and v in the graph. If there are multiple solutions, you may output any one of them.

Examples

standard input	standard output

5 8
0 0
2 0
0 2
2 2
1 1
1 2
1 3
1 5
2 4
2 5
3 4
3 5
4 5
	1
	2
	2
	1
	3

6 10
0 0
1 0
1 1
2 1
0 2
1 2
1 2
1 3
1 5
2 3
2 4
3 4
3 5
3 6
4 6
5 6
	1
	2
	3
	4
	2
	1

Problem I. Ranks

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

A finite field $\mathbf{F}_2$ consists of two elements: 0 and 1. Addition and multiplication on $\mathbf{F}_2$ are those on integers modulo two, as defined below.

$+$	0	1
0	0	1
1	1	0

$\times$	0	1
0	0	0
1	0	1

A set of vectors $v_1, \ldots, v_k$ over $\mathbf{F}_2$ with the same dimension is said to be linearly independent when, for $c_1, \ldots, c_k \in \mathbf{F}_2$, $c_1v_1 + \ldots + c_kv_k = 0$ is equivalent to $c_1 = \ldots = c_k = 0$, where 0 is the zero vector, the vector with all its elements being zero.

The rank of a matrix is the maximum cardinality of its linearly independent sets of column vectors. For example, the rank of the matrix $\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}$ is two; the column vectors $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ and $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ (the first and the third columns) are linearly independent while the set of all three column vectors is not linearly independent. Note that the rank is zero for the zero matrix.

Given the above definition of the rank of matrices, the following may be an intriguing question. How does a modification of an entry in a matrix change the rank of the matrix? To investigate this question, let us suppose that we are given a matrix A over $\mathbf{F}_2$. For any indices i and j, let $A^{(ij)}$ be a matrix equivalent to A except that the (i, j) entry is flipped, i.e. $A^{(ij)}_{kl} = A_{kl} + 1$ for both equations $k = i$ and $l = i$ are held, and $A^{(ij)}_{kl} = A_{kl}$ otherwise.

In this problem, we are interested in the rank of the matrix $A^{(ij)}$. Let us denote the rank of A by r, and that of $A^{(ij)}$ by $r^{(ij)}$. Your task is to determine, for all (i, j) entries, the relation of ranks before and after flipping the entry out of the following possibilities: (1) $r^{(ij)} < r$, (2) $r^{(ij)} = r$, or (3) $r^{(ij)} > r$.

Input

The input consists of a single test case of the following format.

n m

$A_{11} \ldots A_{1m}$

$\ldots$

$A_{n1} \ldots A_{nm}$

n and m are the numbers of rows and columns in the matrix A, respectively ($1 \le n \le 1000$, $1 \le m \le 1000$). In the next n lines, the entries of A are listed without spaces in between. A_{ij} is the entry in the i-th row and j-th column, which is either 0 or 1.

Output

Output n lines, each consisting of m characters. The character in the i-th line at the j-th position must be either '`-`' (minus), '`0`' (zero), or '`+`' (plus). They correspond to the possibilities (1), (2), and (3) in the problem statement respectively.

Examples

standard input	standard output

2 3
001
101
	-0-
	-00

5 4
1111
1000
1000
1000
1000
	0000
	0+++
	0+++
	0+++
	0+++

10 10
1000001001
0000010100
0000100010
0001000001
0010000010
0100000100
1000001000
0000010000
0000100000
0001000001
	000-00000-
	0-00000-00
	00-00000-0
	+00000+000
	00-0000000
	0-00000000
	000-00000-
	0-000-0-00
	00-0-000-0
	+00000+000

1 1 0	+

Problem J. Colorful Tree

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

A tree structure with some colors associated with its vertices and a sequence of commands on it are given. A command is either an update operation or a query on the tree. Each of the update operations changes the color of a specified vertex, without changing the tree structure. Each of the queries asks the number of edges in the minimum connected subgraph of the tree that contains all the vertices of the specified color.

Your task is to find answers of each of the queries, assuming that the commands are performed in the given order.

Input

The input consists of a single test case of the following format.

n
a_1 b_1
...
a_{n-1} b_{n-1}
c_1 ... c_n
m
$command_1$
...
$command_m$

The first line contains an integer n ($2 \le n \le 10^5$), the number of vertices of the tree. The vertices are numbered 1 through n. Each of the following $n-1$ lines contains two integers a_i ($1 \le a_i \le n$) and b_i ($1 \le b_i \le n$), meaning that the i-th edge connects vertices a_i and b_i. It is ensured that all the vertices are connected, that is, the given graph is a tree. The next line contains n integers, c_1 through c_n, where c_j ($1 \le c_j \le 10^5$) is the initial color of vertex j. The next line contains an integer m ($1 \le m \le 10^5$), which indicates the number of commands. Each of the following m lines contains a command in the following format: "`U` x_k y_k" or "`Q` y_k".

When the k-th command starts with '`U`', it means an update operation changing the color of vertex x_k ($1 \le x_k \le n$) to y_k ($1 \le y_k \le 10^5$). When the k-th command starts with '`Q`', it means a query asking the number of edges in the minimum connected subgraph of the tree that contains all the vertices of color y_k ($1 \le y_k \le 10^5$).

Output

For each query, output the number of edges in the minimum connected subgraph of the tree containing all the vertices of the specified color. If the tree doesn't contain any vertex of the specified color, output -1 instead.

Example

standard input	standard output
5 1 2 2 3 3 4 2 5 1 2 1 2 3 11 Q 1 Q 2 Q 3 Q 4 U 5 1 Q 1 U 3 2 Q 1 Q 2 U 5 4 Q 1	2 2 0 -1 3 2 2 0

Problem K. Sixth Sense

Input file:	*standard input*
Output file:	*standard output*
Time limit:	7 seconds
Memory limit:	512 mebibytes

Ms. Future is gifted with precognition. Naturally, she is excellent at some card games since she can correctly foresee every player's actions, except her own. Today, she accepted a challenge from a reckless gambler Mr. Past. They agreed to play a simple two-player trick-taking card game.

Cards for the game have a number printed on one side, leaving the other side blank making indistinguishable from other cards.

A game starts with the same number, say n, of cards being handed out to both players, without revealing the printed number to the opponent.

A game consists of n tricks. In each trick, both players pull one card out of her/his hand. The player pulling out the card with the larger number takes this trick. Because Ms. Future is extremely good at this game, they have agreed to give tricks to Mr. Past when both pull out cards with the same number. Once a card is used, it can never be used later in the same game. The game continues until all the cards in the hands are used up. The objective of the game is to take as many tricks as possible.

Your mission of this problem is to help Ms. Future by providing a computer program to determine the best playing order of the cards in her hand. Since she has the sixth sense, your program can utilize information that is not available to ordinary people before the game.

Input

The input consists of a single test case of the following format.

n

$p_1 \dots p_n$

$f_1 \dots f_n$

n in the first line is the number of tricks, which is an integer between 2 and 5000, inclusive. The second line represents the Mr. Past's playing order of the cards in his hand. In the i-th trick, he will pull out a card with the number p_i ($1 \le i \le n$). The third line represents the Ms. Futures hand. f_i ($1 \le i \le n$) is the number that she will see on the i-th received card from the dealer. Every number in the second or third line is an integer between 1 and 10^4, inclusive. These lines may have duplicate numbers.

Output

The output should be a single line containing n integers $a_1 \dots a_n$ separated by a space, where a_i ($1 \le i \le n$) is the number on the card she should play at the i-th trick for maximizing the number of taken tricks. If there are two or more such sequences of numbers, output the lexicographically greatest one among them.

Examples

standard input	standard output

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

5
3 4 5 6 7
1 3 5 7 9
	9 5 7 3 1

5
3 2 2 1 1
1 1 2 2 3
	1 3 1 2 2

5
3 4 10 3 9
2 7 3 6 9
	9 7 3 6 2