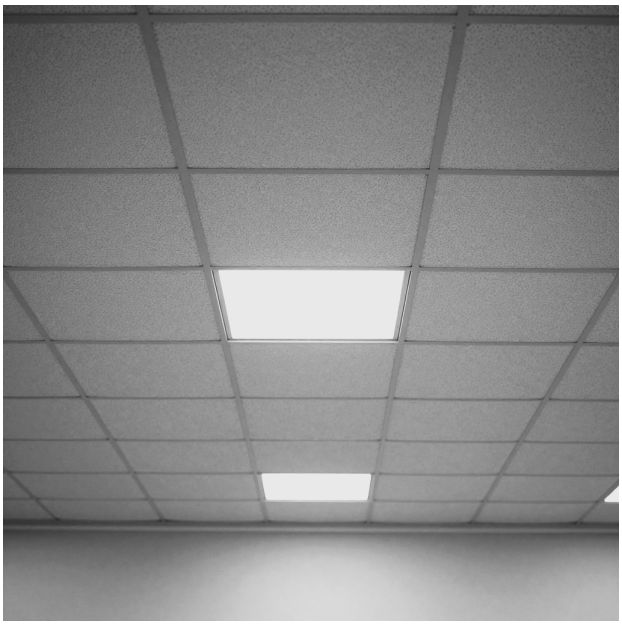
Problem A. Ceiling Tiles

Monocarp decided to set up an office for himself. He has already found and rented a room, and now it is finally time to renovate it.

As in any office, the ceiling will be made as a suspended ceiling of PVC tiles, which can be replaced with square lamps. More precisely, the ceiling is a rectangle consisting of $n \times m$ tiles.



To light his future office, Monocarp wants to replace some of the tiles with lamps. It is impossible to install more than one lamp on the same tile.

At the moment, Monocarp has brought two independent electrical lines to the ceiling, each with its own switch. Each lamp will be connected either to the first line or to the second one.

Monocarp wants his office to be well lit. Namely, he wants to install the lamps in such a way that turning on even one line would leave no *dark places*. Monocarp defines *dark place* as a $k \times k$ square with no lit lamps.

In other words, Monocarp wants to install the lamps in such a way that no matter which line is turned on, there is at least one lit lamp in every $k \times k$ square of tiles.

What is the minimum number of lamps Monocarp needs to install?

Input

The first and only line contains three integers n, m, and k ($2 \le n, m \le 10^9$; $2 \le k \le \min(n, m)$) — the dimensions of the ceiling and the constant k.

Output

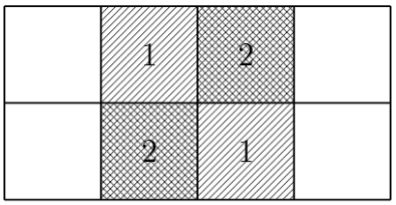
Print one integer — the minimum number of lamps that Monocarp needs to install.

Examples

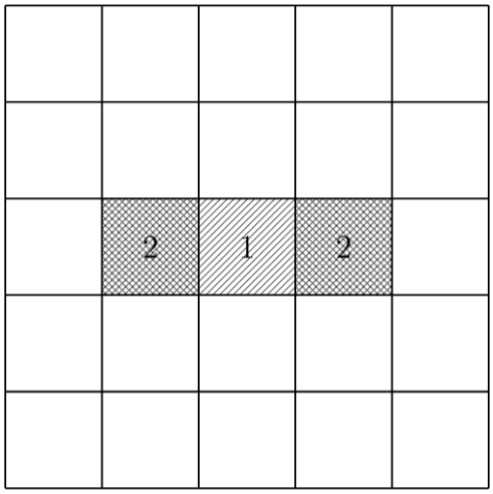
standard input	standard output
2 4 2	4
5 5 3	3
1000000000 999999999 2	499999999000000000

Note

One possible arrangement of lamps for the first example is shown below:



One possible arrangement of lamps for the second example is shown below:



Problem B. Minimizing the Maximum

You are given a string s consisting of n lowercase Latin letters.

You need to distribute all letters from s into k strings $t_1, t_2, \ldots, t_k$. Each letter of the string s must belong to exactly one of the k strings, but some of the strings t_i may remain empty.

Moreover, you may arrange the letters inside each string t_i in any order.

Your task is to minimize the lexicographic maximum among all t_i and output it.

A string a is lexicographically smaller than string b if and only if one of the following holds:

- a is a prefix of b, but $a \neq b$; or
- in the first position where a and b differ, the string a has a letter that appears earlier in the alphabet than the corresponding letter in b.

In particular, this means that the empty string is lexicographically the smallest.

Input

The first line contains two integers n and k ($1 \le n \le 10^6$; $1 \le k \le n$) — the length of the string s and the number of strings which you need to distribute it into.

The second line contains the string s itself, consisting of n lowercase Latin letters.

Output

Print one string — the minimum possible lexicographic maximum among all t_i.

Examples

standard input	standard output
3 2 zzz	zz
7 7 saratov	aaorstv

Problem C. Almost Equal Closures

Let's call the *closure* of a non-empty string s the infinite string $C(s)$ obtained by concatenating s with itself infinitely many times.

For example, $C(\text{x}) = \texttt{xxxxxx}\ldots$, $C(\text{uw}) = \texttt{uwuwuwuwuwu}\ldots$, $C(\text{abac}) = \texttt{abacabacabacabac}\ldots$.

You are given two integers n and m. Your task is to construct two strings s and t such that

1. strings s and t consist of lowercase Latin letters;
2. the length of s is n;
3. the length of t is m;
4. $C(s)$ and $C(t)$ are different, but the first position at which they differ is **as far to the right as possible**.

Input

The first and only line contains two integers n and m ($1 \le n, m \le 10^6$) — the lengths of strings s and t, respectively.

Output

Print two strings s and t satisfying the conditions of the problem. If there are multiple answers, print any of them.

Examples

standard input	standard output
2 4	aa aaab
5 3	babba bab
30 30	abcdefghijklmnopqrstuvwxyzabca abcdefghijklmnopqrstuvwxyzabcb

Note

In the first example, the closures are:

$$\begin{aligned} C(\text{aa}) &= \texttt{aaa}\underline{\texttt{a}}\texttt{aa}\ldots \\ C(\text{aaab}) &= \texttt{aaa}\underline{\texttt{b}}\texttt{aa}\ldots \end{aligned}$$

In the second example, the closures are:

$$\begin{aligned} C(\text{babba}) &= \texttt{babbab}\underline{\texttt{a}}\texttt{bb}\ldots \\ C(\text{bab}) &= \texttt{babbab}\underline{\texttt{b}}\texttt{ab}\ldots \end{aligned}$$

Problem D. Energetic Cook

Monocarp the Cook works in a large restaurant. He can cook two types of dishes: soup and dumplings. He likes to cook soup more — he spends **one unit of energy to cook a soup serving** and **two units of energy to cook a dumpling serving**. Soup and dumplings have different shelf life times:

- a soup serving must be consumed either on the same day it is cooked or on the next day;
- a dumpling serving can be stored for an infinite amount of days, so it can be consumed on any day, starting from the day it is cooked.

The restaurant is open for n consecutive days. During the i-th day, there will be a_i visitors in the restaurant. Monocarp should give either a serving of soup or a serving of dumplings to each visitor. Monocarp can serve any dish at his discretion. He can also serve different dish types during the day. The only requirement is that the dish's shelf life must be within allowed limits:

- if a soup serving is given on day 1, it must have been cooked on the same day;
- if a soup serving is given on day $i > 1$, it must have been cooked on day $i - 1$ or on day i;
- if a dumpling serving is given on day i, it may have been cooked on any day between 1 and i inclusive.

Every day Monocarp starts with the **same** amount of energy E — that is, he can cook any amount of soup and dumplings as long as the total energy he spends during each day is not greater than E. It is impossible to "conserve" energy — for example, if $E = 1$, Monocarp cannot choose to do nothing on one day and cook a serving of dumplings on the next day.

Monocarp would like to know the minimum possible amount of energy E which he needs to have at the beginning of every day (same for each day) that would be enough to feed all visitors if he cooks dishes optimally.

Input

The first line of the input contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of consecutive days the restaurant is open.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$), where a_i is the number of visitors during the i-th day.

Output

Print one integer — the minimum amount of energy Monocarp needs to be able to cook enough dishes for all visitors.

Examples

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

Problem E. Tournament

In a distant galaxy, a space football tournament is being held. All 2^n national teams of star systems take part in the competition. The draw has already been made, and the tournament bracket is known: team 1 will play against team 2, team 3 — against team 4, and so on. Then the losing teams are eliminated and the winning teams play against each other until all teams but one are eliminated. Each time, pairs are formed from the sequence of winners of the previous round, sorted by their indices in increasing order. In particular, the winner of the game between team 1 and team 2 faces the winner of the game between team 3 and team 4, the winner of the game between team 5 and team 6 faces the winner of the game between team 7 and team 8, and so on.

Each team is characterized by two values: a game experience value g_i and a media rating value m_i. The values of g_i of all teams are pairwise distinct. In every match, the team with the higher experience defeats the team with the lower experience — that is, of course, unless the referee interferes.

The tournament organizers want to sell the rights to broadcast it on TV. However, the TV companies want the tournament to be as media-friendly as possible. The total media rating is calculated as the sum of the ratings of the participants of each game. In other words, if a team plays several games, its media rating is counted the same number of times in the total media rating.

The organizers cannot change the structure of the tournament. However, they know that if the referee is given a "bonus" of d space dollars, then the referee will "increase" the game experience of the **weaker** team in the pair by **exactly** d units. If the game experience of the weaker team becomes **strictly greater** than the game experience of the stronger team, then the weaker team wins. Otherwise, the stronger team wins. Note that the game experience of the winning team resets after the match.

The organizers decided to give the referee of each match the **same** bonus (of course, this bonus should be a non-negative integer amount of space dollars). Your task is to determine the minimum possible bonus that maximizes the total media rating of the tournament.

Input

The first line contains one integer n ($2 \le n \le 8$).

The second line contains 2^n integers $g_1, g_2, \ldots, g_{2^n}$ ($1 \le g_i \le 10^9$) — the game experience values of the teams. All values of g_i are pairwise distinct.

The third line contains 2^n integers $m_1, m_2, \ldots, m_{2^n}$ ($1 \le m_i \le 10^9$) — the media rating values of the teams.

Output

Print two integers — the maximum possible total media rating of the tournament and the minimum possible bonus that allows obtaining this rating.

Example

standard input	standard output
3 10 14 37 22 25 18 20 31 32 19 25 20 21 27 35 30	382 16

Problem F. Distances in a Subtree

You are given a tree, consisting of n vertices. The 1-st vertex is the root; for every other vertex i, two integers p_i and w_i are given, which mean that the parent of the i-th vertex is p_i, and the edge connecting these two vertices has weight w_i. The distance between two vertices $d(x, y)$ is equal to the total weight of all edges on the simple path between them.

Each vertex i has a color c_i. You need to process q queries of 2 types:

1. `v x` — recolor vertex v to color x, that is, assign $c_v := x$;
2. `v` — consider all pairs of vertices (x, y) in the subtree of vertex v (including vertex v itself) such that $x \neq y$, $c_x = c_y$, and vertex x is an ancestor of vertex y. For all such pairs, compute the distance $d(x, y)$, and then output the minimum such distance, or -1 if there are no such pairs.

Input

The first line contains two integers n and q ($2 \le n \le 2 \cdot 10^5$; $1 \le q \le 2 \cdot 10^5$).

The next line contains integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le n$), where c_i is the color of vertex i.

The next line contains integers $p_2, p_3, \ldots, p_n$ ($1 \le p_i < i$), where p_i is the parent of vertex i.

The next line contains integers $w_2, w_3, \ldots, w_n$ ($1 \le w_i \le 10^9$), where w_i is the weight of the edge between vertex i and its parent p_i.

The following q lines contain the queries. Each query starts with two integers t_i and v_i ($1 \le t_i \le 2$; $1 \le v_i \le n$) — the type of the query and the index of the vertex it is applied to. If $t_i = 1$, the third integer x_i ($1 \le x_i \le n$) follows.

Output

For each query of the second type, print one integer — the answer to it.

Example

standard input	standard output
5 10 1 1 1 1 1 1 2 2 3 1 1 1 3 2 1 2 2 2 3 2 4 1 2 2 2 1 2 2 1 3 3 2 1 2 2	1 1 3 -1 2 3 2 -1

Problem G. Massage Schedule

Recently, Monocarp was appointed as a director of a bathhouse, and now he faces an unusual task — creating work schedules.

There are n massage therapists working in the bathhouse in total, and the schedule of the i-th therapist is described by one positive integer a_i.

Also, currently m visitors have signed up for massages. For each of them, Monocarp must choose a specific therapist and a day as a pair (i, r), where $1 \le i \le n$ and $0 \le r < a_i$. This means that this visitor is assigned to the i-th therapist and will come to them on days $r, r + a_i, r + 2a_i$, and so on (days are counted from 0).

Thus, a_i is the period with which visitors come to the i-th therapist. No more than one visitor can be assigned to one therapist on the same day.

Monocarp wants to choose schedules for the visitors in such a way that the first day when at least two people come on the same day is as late as possible.

Input

The first line contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of massage therapists.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the therapists' schedules.

The third line contains one integer m ($1 \le m \le \sum a_i$) — the total number of visitors.

Output

Print one integer — the latest possible first day when at least two visitors come. If it is possible to choose schedules so that this never happens, print -1.

Examples

standard input	standard output
2 1 3 4	0
2 3 1 3	-1
3 4 2 4 6	2

Problem H. Iamb and Trochee

One autumn evening, Monocarp felt a slight melancholy and, on a whim, decided to compose some poetry. But, being a true mathematician, he decided to approach this task with complete precision and mathematical rigor.

Monocarp knows n different words in total, and for each word he knows two parameters: the number of syllables in the word and the position of the stressed syllable.

Using the available words, Monocarp wants to build a *rhythmic line*: a sequence of words in which every k-th syllable is stressed, for some integer k. Formally, to determine whether a sequence of words is a rhythmic line, we do the following:

1. first, number all syllables of all words consecutively from left to right, starting from 0;
2. then, if there exist two integers r and k ($0 \le r < k$) such that syllable i is stressed if and only if $i \bmod k = r$, then the sequence is a rhythmic line. Otherwise, it is not.

For example, the line "Twinkle, Twinkle, Little Star" is rhythmic, while "Hello world" is not.

Monocarp can use each word any number of times, but he wants his rhytmic line to feature diverse vocabulary. Help Monocarp determine the maximum number of different words he can use in one rhythmic line.

Input

The first line contains one integer n ($1 \le n \le 20\,000$) — the number of different words Monocarp knows.

Then n lines follow. The i-th of them contains two integers s_i and p_i ($1 \le s_i \le 200$; $1 \le p_i \le s_i$) — the number of syllables in the i-th word and the position of the stressed syllable.

Note that all words in the input are considered pairwise distinct, even if their pairs (s_i, p_i) coincide.

Output

Print one integer — the maximum possible number of different words that Monocarp can fit into one rhythmic line.

Examples

standard input	standard output
4 1 1 2 2 2 1 2 2	4
5 4 3 2 2 4 3 4 3 5 1	4

Note

In the first example, Monocarp can build the following sequence of words: $[3, 1, 2, 4]$, whose rhythmic line can be represented as follows: `Oo|O|oO|oO`.

In the second example, Monocarp can build the following sequence: $[1, 2, 3, 2, 4]$, whose rhythmic line looks like this: `ooOo|oO|ooOo|oO|ooOo`.

Problem I. Character Pumping

Monocarp wants to equip his RPG character for a battle. He has to choose a set of equipment consisting of a sword, a shield, a piece of armor, and a helmet. The inventory contains n items. For each item, the following information is known:

- the type of an item (sword/shield/armor/helmet);
- the bonus to the character's strength when using this item of equipment;
- the combat school for which this item is the most suitable. Any number of items of any type may belong to the same combat school. For example, two different swords can belong to the same school.

Monocarp needs to choose a set of exactly 4 items, one of each type, with the maximum bonus. The *set bonus* is the sum of the bonuses of the equipment items it consists of. The chosen items in the set do not have to be from the same school. However, this is desirable — if all 4 items are chosen from the same school, the *set bonus* is doubled.

Help Monocarp to select a sword, a shield, a piece of armor, and a helmet with the maximum *set bonus*.

Input

The first line contains one integer n ($4 \le n \le 10^5$) — the number of items in the inventory.

Then n lines follow, the i-th of them describes the i-th item as follows: it contains a string t_i, an integer b_i, and a string s_i:

- t_i is the item type, so it is one of the four possible strings: `sword`, `shield`, `armor`, or `helmet`;
- b_i is the strength bonus provided by the item ($1 \le b_i \le 10^6$);
- s_i is the combat school of the item ($1 \le |s_i| \le 10$, s_i consists of lowercase Latin letters only).

Additional constraint on the input: the inventory contains at least one item of each type.

Output

Print one integer — the maximum *set bonus* Monocarp can get from an equipment set consisting of one sword, one shield, one piece of armor, and one helmet.

Examples

standard input	standard output
8 sword 3 cat shield 3 bear sword 2 viper shield 2 viper armor 1 viper helmet 2 manticore helmet 2 viper helmet 1 viper	14
5 shield 6 bear helmet 1 viper armor 1 viper sword 1 viper shield 1 viper	9

Problem J. Chips

There are n cells, numbered from 1 to n. The cells i and $(i + 1)$ are considered adjacent for every $i \in [1, n - 1]$. **No cell is adjacent to itself**.

For each cell, you either place a chip in it or not. If you place a chip in the i-th cell, you get c_i coins.

There are some additional constraints, given by a string of n characters $s = s_1 s_2 \dots s_n$. Every character of this string is either `0` or `1`.

Let f_i be the number of cells adjacent to the i-th cell (again, not including the i-th cell itself) that have a chip. The string s places the following constraints: if s_i is `1`, then f_i must be equal to 1; otherwise (if s_i is `0`), f_i **must not be equal** to 1.

Your goal is to place the chips in such a way that meets all of these constraints and maximizes the number of coins you get.

Input

The first line contains one integer t ($1 \le t \le 10^4$) — the number of test cases.

Each test case consists of three lines. The first line contains one integer n ($2 \le n \le 3 \cdot 10^5$). The second line contains n integers $c_1, c_2, \dots, c_n$ ($1 \le c_i \le 10^9$). The third line contains the string s consisting of exactly n characters; every character of s is either `1` or `0`.

Additional constraint on the input: the sum of n over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, print one integer — the maximum amount of coins you can get (or -1, if it is impossible to meet all of the constraints).

Example

standard input	standard output
5 2 10 20 01 3 10 20 30 111 3 10 20 30 010 5 2 1 10 7 3 11011 5 2 1 10 7 3 11111	10 50 30 18 -1

Note

In the first example, you can place a chip in the 1-st cell.

In the second example, you can place chips in the 2-nd and the 3-rd cells.

In the third example, you can place a chip in the 3-rd cell.

In the fourth example, you can place chips in the 2-nd, the 3-rd, and the 4-th cells.

Problem K. A Game with Dice

Consider a game where each player rolls two regular six-sided dice on each turn. The points that the player receives are calculated as follows: let the points shown on the dice be x and y; if $x \neq y$, then the player receives $(x + y)$ points; otherwise, the player receives $4x$ points.

For example:

- if the first die shows 3 and the second die shows 5, the player receives 8 points;
- if the first die shows 6 and the second die shows 1, the player receives 7 points;
- if both dice show 4, the player receives 16 points.

Monocarp has rolled two dice and claims that the amount of points he should receive is n. You have to check whether this is possible; i. e. find any combination of amounts of points shown on the two dice that results in exactly n points, or report that it is impossible.

Input

The only line contains one integer n ($1 \leq n \leq 36$).

Output

Print two integers x and y ($1 \leq x, y \leq 6$) — the amounts of points on the two dice that result in getting exactly n points. If there are multiple answers, print any of them. If it is impossible to receive exactly n points by rolling two dice according to the rules of the game, print `0 0` instead.

Examples

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

Problem L. Squares in a Square

An integer x is called **square** if there exists an integer y such that $y^2 = x$. For example, $25, 36, 100$ are square integers; $5, 10, 32$ are not.

Consider a matrix consisting of n rows and n columns. It is filled with integers from 1 to n^2 as follows:

- the first row contains integers $1, 2, 3, \ldots, n$ from left to right;
- the second row contains integers $n + 1, n + 2, n + 3, \ldots, 2n$ from left to right;
- and so on.

For example, the matrix for $n = 5$ is as follows:

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{pmatrix}.$$

Your task is to find any **two** distinct square integers in the **last column** of this matrix.

Input

The first line contains one integer t ($1 \le t \le 5 \cdot 10^5$) — the number of test cases.

Each test case consists of one line containing one integer n ($2 \le n \le 2 \cdot 10^9$).

Output

For each test case, print two square integers x and y ($1 \le x, y \le n^2$; $x \ne y$) that both belong to the last column of the matrix. If there are multiple answers, print any of them. If there is no answer, print `0 0` instead.

Example

standard input	standard output
3 5 100 998244353	0 0 100 10000 0 0

Problem M. Binary String Counting 2026

Yes, this is yet another problem about counting binary strings.

You are given a string s consisting of n characters; each character is either `1` or `0`. Also, you are given an integer k.

You can perform **up to** k (possibly zero) operations on string s. Each operation has to be of one of three types:

1. choose an index i from 1 to n and replace s_i with its inverse (that is, if s_i is `1`, replace it with `0`, and vice versa);
2. sort the characters of the string s in non-descending order (assuming that `1` is greater than `0`);
3. sort the characters of the string s in non-ascending order (also assuming that `1` is greater than `0`).

String t is **reachable** if it is possible to obtain it from s by applying no more than k operations. You have to calculate the number of reachable strings t.

Input

The first line contains two integers n and k ($1 \le n \le 150$; $0 \le k \le n$).

The second line contains s — a string consisting of exactly n characters. Each character is either `1` or `0`.

Output

Print one integer — the number of reachable strings. Since the answer can be huge, print it modulo 998244353.

Examples

standard input	standard output
7 1 0001000	10
5 2 10110	22
7 0 0101010	1
7 1 0000000	8
30 30 000000000000000000000000000000	75497471

Note

In the first example, the following strings are reachable: `0001000`, `1001000`, `0101000`, `0011000`, `0000000`, `0001100`, `0001010`, `0001001`, `1000000`, `0000001`.

Problem N. Cheap Sort

You are given an array a of size n, where each integer from 1 to n appears exactly once. The elements of the array are numbered from 1 to n.

To sort the subarray $a[l, r]$ is to reorder all elements $a_l, a_{l+1}, \ldots, a_r$ so that they are in non-decreasing order, without affecting any other elements of the array a. For example, if $a = [3, 5, 2, 1, 7, 4, 6]$ and you sort $a[2, 5]$, you get $[3, 1, 2, 5, 7, 4, 6]$. Sorting the subarray $a[l, r]$ costs $(r - l + 1)$ coins. You can also sort an empty subarray; this costs 0 coins and does not change the array at all.

You want to sort the subarray $a[L, R]$. However, this might be too wasteful, as there might be another subarray $a[L', R']$ such that sorting it yields the same result as sorting $a[L, R]$ but is cheaper.

So, your task is the following: find the minimum possible cost of one sorting operation that yields the same result as sorting $a[L, R]$.

Input

The first line contains one integer t ($1 \le t \le 10^4$) — the number of test cases.

Each test case consists of two lines. The first line contains three integers n, L, R ($1 \le L \le R \le n \le 3 \cdot 10^5$). The second line contains n distinct integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$).

Additional constraint on the input: the sum of n over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, print one integer — the minimum cost of one sorting operation that yields the same result as sorting $a[L, R]$.

Example

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

Note

In the first example, you can sort $a[2, 4]$.

In the second example, you can sort an empty subarray.

In the third example, you have to sort the whole array a (i. e. $a[1, 5]$).