Problem A. AGTC Matrix

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

Let us define a *comparison matrix* for two strings s and t over the alphabet $\{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}$ as follows.

Consider inserting the character '-' into a string. Such a character can be inserted at the beginning or end of the string, as well as between any pair of existing characters in the string.

We will insert an arbitrary number (possibly zero) of such characters into s and t so that their lengths become equal, and then write the resulting strings as the *comparsion matrix* consisting of two rows.

For example, for the strings "ACT" and "AGTT" one of the possible comparison matrices looks like this:

```
A C - T
A G T T
```

The matrices where some column contains two '-' signs are not allowed.

The *value* of the comparison matrix is calculated as the sum of the values of each column. The *value* of a column is determined as follows: if the characters in the column are the same and both are not equal to '-', then the value is $+1$, in all other cases it is equal to -1. For example, the value of the matrix mentioned above is $+1-1-1+1=0$. The largest possible value of a comparison matrix for two strings is called their *similarity*.

A *substring* of the string $s = s_1 \dots s_{|s|}$ is defined as the string $s_{ij} = s_i \dots s_j$ such that $1 \le i \le j \le |s|$. We do not consider empty substrings in this problem. For given two strings s and t, find the substring s' of the string s and substring t' of the string t such that the similarity of s' and t' is maximal among all substrings of s and t.

Input

The first line of the input contains two space-separated integers n and m ($1 \le n, m \le 10^3$), lengths of the strings s and t.

The second line of the input contains the string s, consisting of exactly n characters from the alphabet $\{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}$. The third line of the input contains the string t of length m over the same alphabet.

Output

Print the maximum similarity of strings s' and t', where s' and t' are substrings of the strings s and t, respectively.

Examples

standard input	standard output
9 7 CACCGTAAA TCCGAAA	5
3 3 ACC TGG	-1

Note

For the first sample, the substrings with maximal similarity are "CCGTAAA" and "CCGAAA".

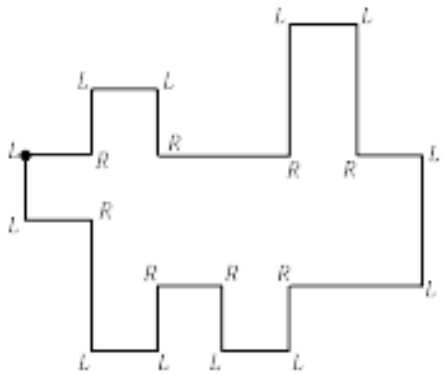
For the second sample, any pair of the characters for two strings have the value -1, and any longer substrings have even lesser similarity.

Problem B. Boundary String

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Consider a polygon P with N vertices and edges parallel to the coordinate axes. The edges of the polygon P can only intersect at their ends, exactly two edges intersect in each vertex, and they meet at a right angle.

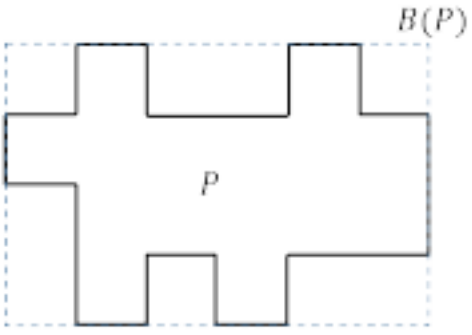
When moving **counterclockwise** along the boundary of polygon P, we turn left or right at each vertex. Let us express it as '`L`' when we turn left at the current vertex, and '`R`' when we turn right. Then, a *boundary string* consisting of '`L`' and '`R`' represents a polygon. For example, the polygon in the figure below is represented by the boundary string "`LLRLLRRLLRLLRLLRRLLR`".



When we represent a polygon by a boundary string, the first letter of the string corresponds to the upper vertex of the leftmost side of the polygon. You can see that this letter is always '`L`'.

Let a *good* polygon be the one that meets the following conditions: the length of each side is an **integer**, and intersection of any vertical line $x = c$ with P contains at most two points that are strictly inside the horizontal edges of the polygon. In this problem, we consider only *good* polygons.

Denote as $B(P)$ the rectangle with the axis-parallel sides containing P such that both vertical and horizontal sides have the minimal length. You can see that this rectangle is formed by the vertical and horizontal lines overlapping the leftmost, rightmost, top and bottom sides of P.



Given a boundary string of length N consisting of letters '`L`' and '`R`', find a *good* polygon P such that the area of $B(P)$ is minimal possible, and print that area.

Input

The first line of the input contains one integer N ($4 \le N \le 800$). The second line contains the boundary

string s consisting of exactly N letters 'L' and 'R'. The first letter of the string is 'L'. You may assume that there exists at least one *good* polygon P such that s is its boundary string.

Output

Print one integer: the minimum possible value of area of $B(P)$.

Examples

standard input	standard output
20 LLRLLRRLLRLLRLLRRLLR	15
10 LLRLLRLLLR	9

Problem C. Convex Shell

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Let us define the *shell* of a convex polyhedron as the set of points outside the polyhedron, with the property that the distance from the point to polyhedron is no more than a given value d.

Given an arbitrary convex polyhedron P and a value d, calculate the volume of the shell. Note that polyhedron may be not strictly convex: it may contain some adjacent faces that are located on the same plane.

Input

The first line of the input contains one integer V: the number of vertices ($1 \le V \le 100$). Then V lines follow, each containing three real numbers x_i, y_i, z_i: coordinates of i-th vertex ($-10 \le x_i, y_i, z_i \le 10$).

Then, after an empty line, follows a line containing one integer E: number of edges of the polyhedron ($1 \le E \le 100$). Then E lines follow, each containing two integers a and b: the zero-based indices of the vertices connected by an edge ($0 \le a, b < V$, $a \ne b$, each pair is listed no more than once).

Then, after an empty line, follows a line containing one integer F: the number of faces ($1 \le F \le 100$). Then F lines follow, each containing a description of one face. The description starts with an integer N, the number of edges in the polygon that form the face, followed by N zero-based indices of the edges e_{ij}, listed in arbitrary order ($0 \le e_{ij} < E$, each edge is listed once for each face it belongs to).

Then, after an empty line, one real number d follows: the parameter of the shell ($0 \le d \le 1$).

All real numbers are given with at most 5 digits after the decimal point.

Output

Print the volume of the shell. Your answer will be considered correct if its absolute or relative error does not exceed 10^{-4}.

Formally, let your answer be a, and the jury's answer be b. Your answer is accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-4}$.

Example

standard input	standard output
8 -1.00000 -1.00000 -1.00000 1.00000 -1.00000 -1.00000 -1.00000 1.00000 -1.00000 1.00000 1.00000 -1.00000 -1.00000 -1.00000 1.00000 1.00000 -1.00000 1.00000 -1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 12 0 1 2 0 0 4 1 5 1 3 2 3 6 2 3 7 5 4 4 6 7 5 6 7 6 4 5 1 0 4 4 8 2 0 3 4 9 2 1 6 4 3 10 4 7 4 11 6 5 7 4 11 9 8 10 0.25	7.24355

Problem D. Determine The Lap Length

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

This is an interactive problem.

You control a racing robot that is placed on a circular runway with integer length in meters. Initially, the robot is placed at the starting point.

The robot accepts the command "`run` k" that orders it to run exactly k meters counterclockwise from its current location and report the number of laps it passed while performing this and all previous commands. In other words, the number of laps is the number of times the robot visited the starting point after the start of the race.

Your task is to determine the lap length using no more than 100 commands.

Interaction Protocol

The interaction is started by your program sending a line with the command "`run` k" (k is an integer, $1 \le k \le 10^9$), which commands the robot to run exactly k meters counterclockwise. Then the jury program prints a line with one integer: the number of laps the robot passed from the beginning after completing this command.

You may assume that the lap length is an integer between 1 and 10^9, inclusively.

You may use at most 100 "`run`" commands.

When you are ready to print the answer, print a line with the command "`ensure` s", where s is the lap length. After printing the answer, terminate your program gracefully.

For correct interaction, print the end-of-line after each command and flush the output buffer with the respective functions of your programming language:

- `cout.flush()` or `fflush(stdout)` for C/C++;
- `stdout.flush()` for Python;
- see documentation for other languages.

Example

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

Problem E. Empires

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

On a planet Byteine in the galaxy far far away there are n cities and m bidirectional roads connecting those cities.

There are three powerful empires controlling those cities. Each empire controls at least one city, and each city is controlled by exactly one empire, that is, the i-th city is controlled by empire e_i (empires are enumerated by integers 1, 2, and 3).

Each Emperor is worried that any of two other Emperors may attack his empire. In this galaxy, military alliances are considered unfair, so if two empires are at war, the third empire keeps a neutral status.

The Emperors thus are planning to set up military bases. Each empire can build military bases only in its own cities. The bases must be placed in such a way that, for any city in the empire, there exists a *safe path* connecting this city to a military base of this empire, regardless of the choice of the enemy empire. A path is considered *safe* if it does not contain any city of the enemy empire (so, it consists of cities controlled by the same empire and cities controlled by the neutral empire).

More formally, for any city of empire A:

- there must be a path connecting this city to a military base of empire A such that it does not contain cities of empire B (for the case of war between empires A and B), and
- there must be a path connecting this city to a military base of empire A such that it does not contain cities of empire C (for the case of war between empires A and C).

The same must be true for the other two empires.

Because setting up the military bases is costly enough, the Emperors ask you to place the minimum number of military bases for each empire.

Input

The first line of the input contains two integers n and m: the number of cities and roads, respectively ($3 \le n \le 30\,000$; $0 \le m \le 10^5$).

The second line contains n integers $e_1, e_2, \ldots, e_n$. Here, e_i is the number of the empire controlling city i ($1 \le e_i \le 3$). There is at least one city in each empire.

Each of the following m lines contains a pair of integers u and v: the cities connected by a road ($1 \le u, v \le n$, $u \ne v$). There is at most one road between every pair of cities.

Output

Print three lines. The i-th line must start with an integer b_i: the minimum number of military bases for the i-th empire. After that, b_i integers must follow: the numbers of cities of the i-th empire where to set up military bases, in any order.

If there are several solutions, print any one of them.

Examples

standard input	standard output
5 4 1 2 1 3 1 1 2 2 3 3 4 4 5	2 1 5 1 2 1 4
8 10 1 1 1 2 2 3 3 3 5 1 5 2 5 3 6 1 6 2 6 3 6 4 7 1 7 4 8 1	1 3 2 4 5 2 8 6

Problem F. Finish Time Expectation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	8 seconds
Memory limit:	512 megabytes

Consider a convex polygon on a plane such that vertices have integer coordinates. The polygon is used to test a new robot. The robot is placed inside the polygon (it can have non-integer coordinates). The robot can travel in any of the four cardinal directions (parallel to coordinate axes) with a constant speed of one unit per second. The robot can change direction instantly at any moment (not necessarily integer).

To test the robot, the engineer randomly and equiprobably selects two points inside the polygon: source and destination (they can have non-integer coordinates). The robot then moves from source to destination in such a way that the movement takes minimal possible time.

Calculate the expected number of seconds it will take for the robot to arrive at the destination.

Input

The first line of the input contains one integer n: the number of vertices of the convex polygon ($3 \le n \le 3 \cdot 10^4$).

The second line contains n space-separated integers: i-th of those integers x_i is the x-coordinate of i-th vertex.

The third line contains n space-separated integers: i-th of those integers y_i is the y-coordinate of i-th vertex.

It is guaranteed that $|x_i|, |y_i| \le 10^9$, the vertices are given in counterclockwise order, and no three vertices of the polygon are collinear.

Output

Print one real number: the expected time. Your answer must have absolute or relative error at most 10^{-9}.

Examples

standard input	standard output
4 0 4 4 0 0 0 2 2	2
4 0 1 0 -1 -1 0 1 0	0.9333333333
3 0 1 3 3 0 2	1.4222222222
5 1 4 6 4 0 0 1 4 5 2	2.8782787990904536169885

Problem G. Generate Optimal Tree

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

There is a set of N binary strings, all of the same length. They are denoted as $s_0, s_1, \ldots, s_{N-1}$. Some string t from this set is selected secretly, and you have to guess this string. In order to do that, you can ask, for any position k in the secret string, what is the digit at that position (the position is zero-based, so $0 \le k < |t|$).

Your task is to build a decision tree that corresponds to the optimal strategy of queries needed to determine the index of the selected string. A strategy is optimal if it performs the least number of comparisons in the worst case.

Each non-leaf node of the decision tree contains an integer k ($0 \le k < |t|$): the position to ask about. The node then has exactly two children: the first is for the case when k-th digit of the secret string is '0', and the second is for the case when k-th digit is '1'.

The leaves of a decision tree correspond to terminal states of the search. Each leaf contains the number of the secret string in the initial list (from 0 to $N - 1$).

Input

The first line of the input contains one integer N ($2 \le N \le 16$).

Then N lines follow, containing strings $s_0, s_1, \ldots, s_{N-1}$. Each of these strings consists of characters '0' and '1'. The strings are pairwise distinct and have the same length $|t|$ ($1 \le |t| \le 32$).

Output

Print the decision tree starting from the root node. Each node is described on a separate line.

Description of a leaf node contains one character '=' followed by a space and a zero-based index i of the secret string in the initial list ($0 \le i < N$).

Description of a non-leaf node contains one character '>' followed by a space and a zero-based index k of the position to check ($0 \le k < |t|$). Then follows the description of the subtree for the case when k-th digit is '0'. After that follows the description of the subtree for the case when k-th digit is '1'.

If there are several possible optimal trees, print any one of them.

Examples

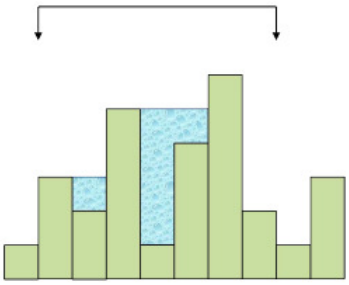
standard input	standard output
5 0010111000 0111101101 0110110001 0011100101 0010111101	> 9 = 0 > 6 > 7 = 2 = 3 > 5 = 1 = 4
3 000 111 110	> 2 > 1 = 0 = 2 = 1

Problem H. Heavy Rain

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	512 megabytes

The surface of some extraterrestrial planet is mostly desert, so it doesn't rain very much. All plants living on this planet are cacti. Because it doesn't rain much, cacti have evolved and have a way to work together to store water.

Imagine some desert on the planet as a two-dimensional plane with only the vertical and horizontal directions. There are N cacti in the area. Each cactus is exactly 1 meter wide. Cacti can have different heights. Sometimes it rained heavily at interval between S-th cactus and the E-th cactus, and the water was collected in the area above the cacti that could contain water. In other words, water is collected as shown in the picture below.



Cacti outside the range indicated by the arrows are not affected by the rain, so they cannot collect the water this time.

There are Q rains, each at its range. For each rain, calculate the amount of water that cacti has accumulated after that rain. The rains are rare enough, so at the time of each rain there is no water accumulated after any previous rains.

Input

The first line of the input contains two integers N and Q: the number of cacti and the number of rains, respectively ($1 \le N, Q \le 5 \cdot 10^5$). Then one line follows, containing N positive integers h_i: the heights of the cacti from left to right, respectively ($1 \le h_i \le 10^9$). Then Q lines follow, each containing two integers S and E: leftmost and rightmost (inclusively) cacti for the respective rain ($1 \le S \le E \le N$).

Output

For each rain, print one integer: the amount of water collected by the cacti after that rain.

Example

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

Problem I. Improved Werewolf

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

The Werewolf is a social deduction game, created in 1986. The game models a conflict between two groups: an informed minority (the werewolves), and an uninformed majority (the villagers). At the start of the game, each player is secretly assigned a role affiliated with one of these teams. The game has two alternating phases: first, a night phase, during which werewolves with night killing powers may covertly kill other players, and second, a day phase, in which surviving players debate the identities of players and vote to kill a suspect.

The game continues until a faction achieves its win condition.

There are some improved versions of that game. Consider the following modification of the game: there is ability to resurrect killed players (regardless of phase they were killed at) and there is a *trust score* implemented that somehow affects the gameplay.

A total of n players are sitting in a row and playing the improved version of Werewolf. There are four types of events:

1. For all players that are still alive, starting from l-th from the left and ending with r-th from the left, inclusively, pick each k-th player (that is, players with numbers $number_\alpha = l + \alpha k$, $number_\alpha \le r$, where $\alpha = 0, 1, 2, \ldots$), and add an integer x to their trust scores.
2. For all players alive, from l-th to r-th, inclusively, pick each k-th player (defined as in the previous action), find the minimum value of trust score among them, and print that value.
3. Kill the l-th player from the left among those who are still alive.
4. The player with initial number d of the n players is resurrected (you may assume that this player is currently killed) with a fresh trust score of 0.

At the beginning of the game, all players are alive and have the trust score 0.

Your task is to write a game engine that will process those events efficiently.

Input

The first line of the input contains two integers n and q: the number of players and the number of events, respectively ($1 \le n, q \le 10^5$). Then q lines follow, each describes one event.

- If this is type 1 event, the description starts with the word "`change`" which is followed by four space-separated integers l, r, k and x.
- If this is type 2 event, the description starts with the word "`getmin`" which is followed by four space-separated integers l, r and k.
- If this is type 3 event, the description starts with the word "`kill`" which is followed by the integer l.
- If this is type 4 event, the description starts with the word "`revive`" which is followed by the integer d.

Denote the number of type 3 events before the current event as e_{killed}, and the number of type 4 events before the current event as $e_{revived}$. Then *alive* is defined as $n - e_{killed} + e_{revived}$, and $1 \le l, r \le alive \le n$; $k \in \{2, 3\}$; $0 \le x \le 10^5$; $1 \le d \le n$.

Output

For each event of type 2, print one integer: the minimum value of the trust score among the requested players.

Examples

standard input	standard output
5 5 change 1 5 2 100 change 2 5 2 10 getmin 1 4 3 kill 4 getmin 1 4 3	10 100
5 5 kill 3 change 1 4 2 1 change 2 4 2 2 revive 3 getmin 1 5 2	0

Problem J. Juggle Sort

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

There are n cards on the table, with integers $a_1, a_2, \ldots, a_n$ written on them from left to right.

We start from position 1 and will repeat the following process. Let the the current position be i. If there is $j > i$ such that $a_j > a_i$, then we are pay 1 coin and move the card from the current position to the right end (in other words, we put i-th card to the n-th position, and cards at positions $i + 1$ to n are moved one position to the left). Otherwise, the current position becomes $i + 1$.

The process stops when we reach position n.

Given initial values of a_i, calculate how many coins you need to pay to finish the process.

Input

The first line of the input contains one integer n: the number of cards on the table ($1 \le n \le 10^6$).

The second line contains n space-separated integers $a_1, a_2, \ldots, a_n$: the initial values of the cards on the board, from leftmost one to rightmost one ($1 \le a_i \le 10^9$).

Output

Print one integer: the number of coins you need to pay to finish the process.

Example

standard input	standard output
10 3 7 6 8 5 8 2 1 7 6	14

Note

- Consider an example **3 7 6 8 5 8 2 1 7 6**. When we are going from left to right, you may see that cards **3 7 6 5 2 1** (at the positions 1 2 3 5 7 8) have to their right a card with greater value. So they will be moved to the right for one coin each. After those cards will be moved, we have the order **8 8 7 6 3 7 6 5 2 1** where we are currently at position 4.
- With that order, we will move the cards **6** and **3** (placed at the positions 4 and 5 now), because there is card **7** to their right, and will obtain **8 8 7 7 6 5 2 1 6 3** with current position 5.
- Then the cards **5 2 1** (from the positions 6 7 8) are moved, obtaining **8 8 7 7 6 6 3 5 2 1** with current position 6.
- Then the card **3** from the position 7 is moved, obtaining **8 8 7 7 6 6 5 2 1 3** with current position 8.
- Then finally cards **2 1** (positions 8 9) are moved, so we obtain the order **8 8 7 7 6 6 5 3 2 1** with the final position reached.

In total, we paid $6 + 2 + 3 + 1 + 2 = 14$ coins.

Problem K. King And Toll Roads

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

There are N cities and $M + N - 1$ two-way roads along the Byteotian river. The cities are enumerated by integers from 1 to N. For each i such that $1 \leq i < N$, there is a road connecting cities i and $i+1$. The other M roads connect some other pairs, so that each pair of the cities is directly connected by at most one road.

There are $N - 1$ cars in each city each morning, so that there is one car going from every city to every other city. Formally, for each pair of distinct cities u and v, there is exactly one car starting in u and finishing in v, and there is exactly one car starting in v and finishing in u. By the evening, all cars arrive at their destinations.

The King wants to select exactly two roads and set the toll of 1 bytaler for driving along each of those roads (so if some car goes along both roads, the toll for this car is 2 bytalers). The drivers know which roads are selected for the toll, so they choose the route with the least possible toll while driving between starting and target city.

Your task is to select two toll roads in such a way that the total payment received in one day is maximized.

Input

The first line of the input contains two integers N and M: the number of cities and the number of non-trivial roads, respectively ($3 \leq N \leq 5 \cdot 10^5$; $0 \leq M \leq 5 \cdot 10^5$). Each of the following M lines contains two integers A and B: the numbers of the cities connected by a non-trivial road ($|A - B| > 1$, $1 \leq A, B \leq N$). Each pair of cities is directly connected by at most one road.

Output

Print one integer: the maximum total payment received in one day.

Examples

standard input	standard output
4 3 3 1 2 4 1 4	0
4 1 1 3	8
4 0	14

Problem L. LTE Broadcasting Stations

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

There are N LTE broadcasting stations, $s_1, s_2, \ldots, s_N$, located on a straight line. The coordinate of station s_i is given by a positive integer x_i, and you must allocate a non-negative integer r_i as the broadcast range for each station s_i. If the station s_i broadcasts, stations within distance of r_i from s_i can receive broadcasts of s_i. In other words, if the LTE broadcasting station s_j is located within the closed segment $[x_i - r_i, x_i + r_i]$, then s_j can receive the signal from s_i.

Signal of LTE station s_i can be delivered to LTE station s_j by successively delivering through $h-1$ ($h > 1$) broadcasting stations. Formally, if there are broadcasting stations $s_{i_1}, s_{i_2}, \ldots, s_{i_{h-1}}$ such that signal of s_i is received by s_{i_1}, signal of s_{i_k} is received by $s_{i_{k+1}}$ and signal of $s_{i_{h-1}}$ is received by s_j, we will say that signal from s_i is delivered to s_j in h hops. In the special case of $h = 1$, the broadcast of s_i shall be directly delivered to s_j.

When we allocate the broadcast range r_i to each LTE station s_i, the total allocation cost is defined as $\sum_{i=1}^{N} r_i \cdot \lfloor \sqrt{r_i} \rfloor$. We will allocate the broadcast ranges to minimize this cost.

We will call the LTE station s an *h-central station* if it can receive signal from any of other $N-1$ stations in h or less hops. Let $C_h(s)$ be the minimum cost of allocating the broadcast ranges to the $N-1$ stations so that there is at least one h-central station.

For example, let 5 broadcasting stations $s_1, \ldots, s_5$ be located at coordinates 1, 3, 4, 6 and 9, respectively, and $h = 2$. If we choose s_3 as the 2-central station, we can allocate broadcasting ranges 2, 1, 2, and 3 to s_1, s_2, s_4, and s_5, respectively, and the total allocation cost is 8. You may check that 8 is the lowest possible cost for $h = 2$, so $C_2(s) = 8$.

Given the locations of N stations, for each $h = 1, 2, \ldots, N-1$, print the value of $C_h(s)$.

Input

The first line of the input contains one integer N: the number of stations ($2 \le N \le 120$). The second line contains N integers $x_1, x_2, \ldots, x_n$ ($1 \le x_1 < x_2 < \ldots < x_n \le 10^8$).

Output

Print $N-1$ integers: i-th of them must be the value of $C_i(s)$.

Examples

standard input	standard output
3 1 3 8	12 12
5 1 3 4 6 9	16 8 8 8