Problem A. Another LCA Problem

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 mebibytes

Of course you all are familiar with the famous LCA problem. Now you are asked to solve its dynamic version.

Let us recall the problem itself. You are given a rooted tree. It means that each vertex except one (that one is called the *root* of the tree) has some other vertex as a parent. Each vertex is a descendant (that is, direct or indirect child) of the root vertex. Vertex u is called an *ancestor* of vertex v if and only if u coincides with v, u is the parent of v or u is an ancestor of the parent of v. Vertex w is a *common ancestor* of vertices u and v if and only if w is an ancestor of v and w is an ancestor of u. Vertex w is the *least common ancestor* (LCA) of vertices u and v if and only if w is a common ancestor of u and v and no other common ancestor of u and v has w as its parent. It can easily be seen that there is only one least common ancestor for any two vertices.

The LCA problem itself is as follows. Given a tree and series of queries (pairs of vertices), find the LCA for each of the given pairs.

You have to solve the dynamic version of this problem. The queries are the same, but now, the tree itself can be changed between queries. For simplicity, we will consider only one type of change: changing the parent of a vertex.

Input

On the first line of input, there is single integer n, the number of vertices in the tree ($1 \le n \le 10^5$). The vertices are conveniently numbered by integers from 1 to n. On the second line, there are $(n-1)$ integers $p_2, p_3, \ldots, p_n$. Here, vertex 1 is the root of the tree and each vertex $i > 1$ has vertex p_i as its parent. It is guaranteed that the input data forms a valid rooted tree.

The third line contains one integer m, the total number of queries and changes ($1 \le m \le 10^5$). Each of the following m lines contains a description of a query or a tree change request. A query starts with the letter '`Q`' followed by a pair of integers u and v, the number of vertices for which you have to find their least common ancestor. A tree change request starts with the letter '`C`' followed by two integers u and v which mean that the new parent of vertex u will be vertex v. It is guaranteed that all changes are valid in the sense that, after each change, vertex 1 is still the ancestor of all other vertices.

Output

For each query, print one integer on a line: the least common ancestor of the specified pair of vertices.

Examples

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

Problem B. Bar

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	256 mebibytes

An infinite number of mathematicians walk into a bar. The first goes up to the bartender and says, "I'll have a pint of lager, please." Each next one says, "and I'll have half of what he's having." The bartender says, "You're all idiots," and pulls two pints.

It's a well-known joke, but anyone hardly knows that this is the real story. Moreover, it is the usual situation in the Berland bar.

Every day, an infinite number of students of mathematics department of Berland university walk into a bar. And since the stocks of alcohol are limited, they have to construct convergent series by their orders.

Berland mathematicians are very friendly. So recently they decided that it is unfair to order $\frac{1}{2^{n-1}}$ of pint because the general term of this series converges very fast.

So now n-th mathematician is ordering $\frac{\ln n}{n^s}$ of pint.

The bartender is your very good friend. He cannot calculate the sum of this series, so he asks you to help him.

Input

The first line of input contains an integer n ($1 \le n \le 10^5$), the number of test cases. Each of the next n lines contains one real number s ($1.0001 \le s \le 20$). This number is given with at most eight digits after decimal point.

Output

For each test case, output a single number on a separate line. Absolute error must be not greater than 10^{-6} of the infinite sum $\sum_{n=1}^{\infty} \frac{\ln n}{n^s}$.

Example

standard input	standard output
3 2 3 4	0.93754824 0.19812624 0.06891127

Problem C. Counting Palindromes

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

A palindrome is a character string that may be read the same way both from left to right and from right to left. Palindromology is a science that searches for palindromes wherever they can be found. Today the given string S consisting of lowercase Latin letters is being searched.

Let $S[i, j]$, $i \leq j$ denote a substring from character i to j. The task is to calculate the number of substring-palindromes for substring $S[a, b]$. In other words, you need to find out the amount of $S[x, y]$, such that:

- $a \leq x \leq y \leq b$,
- $S[x, y]$ is a palindrome.

Input

The first line contains input string S, the length of which does not exceed 10^5 symbols. The next line contains the number of requests m ($1 \leq m \leq 3 \cdot 10^5$). Each of the following m lines consists of a pair of integers a and b ($1 \leq a \leq b \leq n$, where n is the length of string S), which denote a request for searching for substring-palindromes in substring $S[a, b]$.

Output

For each request output the answer on a single line.

Examples

standard input	standard output
abaa 4 1 2 1 3 1 4 3 4	2 4 6 3

Problem D. Drawing Triangles

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Recently, Petya has found a way not to get bored during a lecture. He draws a triangle and splits it into some number of smaller triangles which are similar to the initial triangle. Petya wishes all the smaller triangles to have pairwise different areas.

Vasya noticed that this task is very complex and Petya should better write a program to construct the partition in the proper way. But Petya is not skilled enough to write such program. That's why he decided to ask you for help.

You are to write a program that constructs the desired partition. Recall that one triangle is similar to another if their sets of angle values are the same. A set of triangles is a partition of the given triangle if each point of the given triangle is covered either by an interior point of exactly one of the triangles in the set or by at least one point on a border of some of the triangles in the set, and no points of the triangles in the set lie outside the given triangle.

Input

The only line of input contains six integers X_1, Y_1, X_2, Y_2, X_3 and Y_3: the coordinates of triangle's vertices. Each of the coordinates doesn't exceed 100 by absolute value. It is guaranteed that the given triangle is non-degenerate.

Output

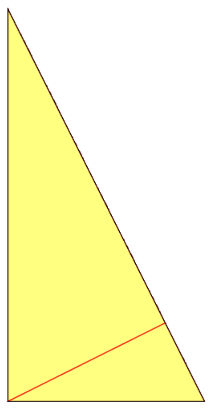
On the first line of output, print an integer N ($2 \le N \le 2000$), the number of triangles in the partition.

On the next N lines, print the descriptions of triangles, one per line; i-th of these lines should contain the coordinates of the i-th triangle's vertices in any order. All triangles should be similar to the given one and should have different areas. Output the coordinates as precise as possible! The checking program has tolerance of 10^{-5} when comparing products of side lengths and triangle areas.

It is guaranteed that the answer always exists with the given constraints.

Example

standard input	standard output	Notes

0 0 0 10 5 0	2
	0.0 10.0 4.0 2.0 0.0 0.0
	0.0 0.0 4.0 2.0 5.0 0.0
		

Problem E. European Football Championship

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

European Football Championship is held every four years since 1960. Since then, the scheme of this tournament has changed many times. The scheme currently used is similar to the one described below, but differs in minor aspects. Please use this statement as a formal and complete document and make no assumptions based on the real-world scheme.

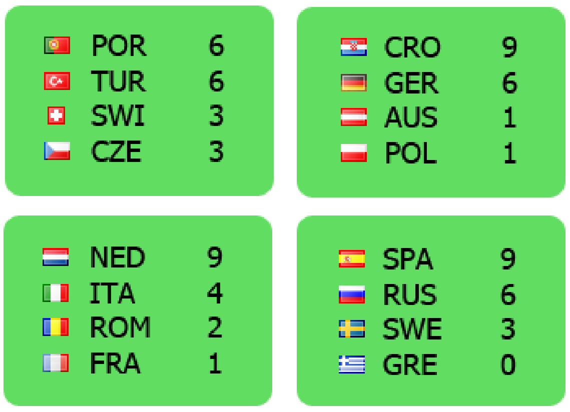
16 teams take part in the final part of the competition. They are divided into four groups of four teams. In each group, each pair of teams plays one match. The points are awarded as follows: three for a win, one for a draw, and none for a loss. After that, teams in a group are ordered according to the number of points. Teams with more points get higher places. In this problem, tie-breaking rule is simplified. If two teams have the same number of points, the team that has the bigger goal difference (the net difference between goals scored and goals conceded) takes the higher place. If teams have the same number of points and the same goal difference, the team with lexicographically smaller three-letter country code takes the higher place.

Then the play-off stage follows. Two best teams from each group take part in this stage. The rules of play-off periodically change, so in this problem, we use a generalized form of the rules. At this stage, draws are impossible. If there is no winner at the end of the game, extra time will be added. In case of the same situation at the end of the extra time, the penalty shoot-out decides who is a winner. The winner of the shoot-out is awarded with one more goal to the result and moves to the next phase of play-off (or become the winner of the tournament if the game is final).

The play-off stage is carried out in three phases: quarterfinal, semifinal, final. At the quarterfinal phase, eight teams are divided into four pairs. Pairs are chosen uniformly at random in such a way that each team that had first place in a group plays one game against a team that had second place in a group. Note that each first-placed team can meet each second-placed team, even if they met at the group stage. At the next phase of semifinal there are only four teams. They are randomly divided into two pairs. Each pair plays the game. The winner from each game advances to the final phase (the final of the tournament). The winner of the final is the European Champion. So the European Football Championship consists of 31 game, including 24 games of the group stage and 7 games of the play-off stage.

Pasha is a big fan of football and, of course, he paid attention to such a big tournament. Once the Championship was over, he recalled and enumerated all the games from it in some arbitrary order. Pasha remembered all teams and results of games, but before the announcement of this list, he wants to make sure that at least one correct scheme of the tournament exists for this list of games.

For example, for the list of games from the sample input, the following scheme of the tournament exists:



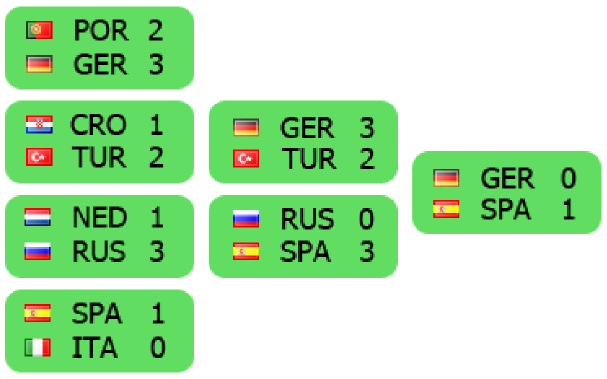
POR	6
TUR	6
SWI	3
CZE	3

CRO	9
GER	6
AUS	1
POL	1

NED	9
ITA	4
ROM	2
FRA	1

SPA	9
RUS	6
SWE	3
GRE	0

Possible group-stage scheme for the sample input



Possible play-off scheme for the sample input

Your task is to write a program that, for a given list of games, defines how many different schemes of tournament exist for this list. Two schemes are considered the same if they have the same compounds of groups, the same results for the group-stage games, opponents are distributed in the same way at each phase of the play-off stage, and all games of this stage have the same results. Otherwise, the two schemes are considered different.

Input

The first line of input contains one positive integer T — the number of test cases ($1 \leq T \leq 1000$).

Then T test cases follow, each of them consists of 31 lines. Each line denotes a single match and has the format `TM1[space]A:B[space]TM2`, here `TM1` and `TM2` are three-letter country codes of teams, `A` and `B` are amounts of scored goals for respective teams ($0 \leq$ `A`, `B` ≤ 9). You may assume that three-letter country codes contain exactly three capital English letters. Each team is uniquely identified by its country code. There are no more than 16 different country codes in each test case.

Output

For each test case, output one of the answers on a single line:

"`There are no possible championships for list #`N`!`" — if for the N-th test case there is no possible championship scheme.

"`There is 1 possible championship for list #`N`.`" — if for the N-th test case there is only one possible championship scheme.

"`There are K possible championships for list #`N`.`" — if for the N-th test case there are $K > 1$ possible championship schemes.

Output should not contain empty lines or extra spaces. Use the format shown in the sample below.

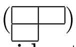
Example

standard input	standard output

`1`
`SWI 0:1 CZE`
`POR 2:0 TUR`
`CZE 1:3 POR`
`SWI 1:2 TUR`
`SWI 2:0 POR`
`TUR 3:2 CZE`
`AUS 0:1 CRO`
`GER 2:0 POL`
`CRO 2:1 GER`
`AUS 1:1 POL`
`AUS 0:1 GER`
`POL 0:1 CRO`
`ROM 0:0 FRA`
`NED 3:0 ITA`
`ITA 1:1 ROM`
`NED 4:1 FRA`
`NED 2:0 ROM`
`FRA 0:2 ITA`
`SPA 4:1 RUS`
`GRE 0:2 SWE`
`SWE 1:2 SPA`
`GRE 0:1 RUS`
`GRE 1:2 SPA`
`RUS 2:0 SWE`
`POR 2:3 GER`
`CRO 1:2 TUR`
`NED 1:3 RUS`
`SPA 1:0 ITA`
`GER 3:2 TUR`
`RUS 0:3 SPA`
`GER 0:1 SPA`
	`There are 2 possible championships for list #1.`

Problem F. Filling Out

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Consider a rectangle consisting of $W \times H$ square cells. This rectangle may contain at most one cut out cell. You must fill it out with L-shaped tiles (). Tiles should not overlap with each other, come out of the rectangle or cover the cut out cell. Every cell besides the one which is cut out should be covered with exactly one tile.

Input

The first line contains two integer numbers W and H $(1 \le W, H \le 1000)$, which are width and height of rectangle. The second line contains two integer numbers X $(0 \le X \le W)$ and Y $(0 \le Y \le H)$, which are coordinates of cut out cell. Left bottom corner has coordinate $(1, 1)$, while right top one has coordinates (W, H). If at least one of X and Y is zero, there is no cut out cell in the rectangle.

Output

Print any appropriate filling of rectangle. Use digits to mark tiles and symbol '`@`' to mark the cut out cell. Two cells covered with the same tile should be marked with the same digit, while two cell with common edge covered with different tiles should be marked with different digits. If there is no appropriate filling, print the word "`Impossible`" instead.

Examples

standard input	standard output

`7 4`
`2 3`
	`1122311`
	`1@23381`
	`3341885`
	`3441155`

`4 5`
`0 0`
	`Impossible`

Problem G. Graph Theory Problem

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Does your research work have any practical applications?

Frequently asked question

Foreword

Serezha has almost finished his thesis. The subject hasn't changed since December: "Dynamic 2-Edge-Connectivity Problem". The algorithm has been invented and tested, the bound of $O(K \log K)$ has been proved. The only thing left is to write about "practical applications".

Well, does the problem "Dynamic 2-Edge-Connectivity Problem" indeed have any practical applications? That's not a simple question. Probably, it's even harder than the problem itself. Whatever, the thesis has to be done.

So, the first practical application: let us create a contest problem about it!

Problem

Given an undirected graph with no more than 10^5 vertices. Initially it does not contain any edges. You have to process requests `ADD x y` and `DEL x y` "— to add and to remove edge from x to y, respectively.

After each request, you should find **the number of bridges** in the graph.

There are no multiedges and loops.

For every request to remove an edge, the corresponding edge exists.

Solution

The thesis has to be pretty hard to be written in five hours. So you are given five hints.

1. Requests to add or remove edge make the edge "alive" during some intervals of time.
2. Use "Divide and conquer" idea.
3. Compress components of biconnectivity.
4. Even having compressed biconnectivity components, the graph can be reduced provided there are few requests.
5. The solution in $O(K \log K)$ exists.

Input

The first line of input contains two integers N and K: the number of vertices and requests, respectively. $1 \le N \le 10^5$, $1 \le K \le 10^5$.

The following K lines contain requests, one per line. Each request starts with a word "`ADD`" or "`DEL`", depending on the type of the request. Two integers a_i and b_i follow, describing the edge to add or to remove. $1 \le a_i, b_i \le N$, $a_i \ne b_i$.

Output

Write K integers: the number of bridges in the graph after each request.

Example

standard input	standard output
4 8 ADD 1 2 ADD 2 3 ADD 1 3 DEL 2 3 DEL 1 2 ADD 2 4 ADD 1 4 ADD 2 3	1 2 0 2 1 2 3 0

Problem H. Hatto Nero

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

The famous mafiosi Hatto Nero and his gang are in trouble. They successfully robbed the Federal Bank of Catland and are now planning the way to escape. There are N cats in the gang and Hatto Nero found N hideouts in different cities for them (he does not need a hideout for himself because he did not take part in the robbery). The gang leader decided that each cat will go to different hideout, so that there will be exactly one cat in each hideout. For safety, the gang selected some roads in the country in such a way that there is exactly one simple path between every pair of hideouts and between the bank and each hideout.

However, there is one problem left: on every road (even on roads selected by the gang) there is a customs official who is of course informed about every member of the gang and will not let them pass. Luckily, the officials are very greedy, so the gang members can *bribe* them: for some amount of money, an official can become "blind" for a short period of time which is enough for one gang member to travel by the road controlled by that official. In order for several gang members to travel by one road, they should all separately bribe the official controlling that road.

Each gang member starts at the bank and moves along the simple path to the hideout assigned to him. For each road the gang member travels by, he has to bribe the official on that road exactly once. Briberies are performed in the order the gang member moves along his path.

Each official i has *rank* b_i and *greediness* c_i. The rank denotes importance of the official in the bureaucratic hierarchy. Each official hates all officials with rank less than his own rank. Because of that, an official can be bribed by a gang member only if that official gets strictly more money than at least half of the officials which were already bribed by this gang member and have strictly lower rank. In case there are no such officials, he will be satisfied with the amount of money equal to his greediness instead. A formal definition follows.

Suppose a gang member already bribed a set U of officials and is now trying to bribe official i. Let $A \subseteq U$ be the set of officials already bribed by this gang member which have rank strictly lower than b_i. If this set A is empty, official i can be bribed for the amount of money equal to his greediness c_i. Otherwise, let $m_1 \leq m_2 \leq \ldots \leq m_{|A|}$ be the amounts of money paid to the officials from set A in non-decreasing order. The official i can then be bribed if the amount of money paid to him is strictly greater than m_k where $k = \left\lceil \frac{|A|}{2} \right\rceil$. The amount of money must be an integer. In the latter case, greediness c_i does not matter.

Hatto Nero is tired after the robbery, so he asked you to find how much money will each member of the gang need to spend on bribing officials.

Input

The first line of input contains the number of gang members N ($1 \leq N \leq 10^5$). The following N lines contain the descriptions of the roads. Each description consists of four integers s_i, t_i, b_i and c_i, where s_i and t_i are numbers of objects connected by the road (the bank has number 0, hideouts are numbered from 1 to N) while b_i and c_i are rank and greediness of the official controlling this road ($0 \leq b_i, c_i \leq 10^9$). All roads are bidirectional.

Output

On the first line of output, print N integers: for each hideout from 1 to N in increasing order, print the amount of money which will be spent on bribing officials by the gang member going to this hideout.

Example

standard input	standard output

8
0 1 1 2
1 3 0 3
1 4 2 4
4 7 3 10
4 8 0 10
0 2 10 4
2 5 8 10
2 6 11 10
	2 4 5 5 14 9 8 15

Problem I. Invasion

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Aliens prepare to invade the Earth! They have built numerous force fields around the Solar system. Each force field can be viewed as an infinite plane in space. Luckily, Earth scientists developed a powerful weapon: Yamato gun. A shot from this weapon flies along a straight line destroying every force field it crosses. Unfortunately, the weapon requires too much energy, so only one shot can be made. The gun can shoot from Earth in any direction.

Your task is to determine the maximum number of force fields which can be destroyed in a single shot.

Input

The first line of the input contains a single integer N ($3 \le N \le 1111$), the number of force fields. Each of N following lines contain six numbers x_0, y_0, z_0, x_d, y_d, z_d. Here, (x_0, y_0, z_0) is a point which belongs to the force field, and (x_d, y_d, z_d) is the normal vector of the force field plane. All coordinates are integers and do not exceed 10^4 by absolute value. The coordinates are given relative to the Earth (the Earth is located in the point of the origin). Every three planes have exactly one common point. No plane crosses the Earth.

Output

On the first line of output, print one integer: the maximal number of force fields which can be destroyed in a single shot.

Example

standard input	standard output

3
1 0 0 1 0 0
0 1 0 0 1 0
0 0 1 0 0 1
	3

Problem J. Jailing Rabbits

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	256 mebibytes

Pasha has a nice garden in the countryside. But nowadays his garden is threatened by insidious rabbits. They steal vegetables and tramp down the garden beds.

Today is the day of revenge! Pasha thoroughly studied the behavior of rabbits, and he has a plan how to protect his garden from the dangerous animals.

Pasha's garden can be represented as a regular grid of size $N \times M$. Each node is either empty or contains a rabbit hole. In each hole live exactly four rabbits. Pasha placed traps for rabbits in each of the corners of the garden (that is, in the following nodes of the grid: $[0, 0]$, $[0, M]$, $[N, 0]$ and $[N, M]$).

Pasha has a paintball gun and K paintballs where K is the number of rabbit holes. All paintballs have different colors which are numbered from 1 to K. Pasha can shoot a paintball at any rabbit hole. After the shot, all the rabbits run out of the hole colored in the color of the paintball. All four rabbits run with the same speed in different directions. Rabbits can only run along grid segments.

While moving, every rabbit leaves footprints of his color everywhere: in the nodes and on the segments they visit. However, at every node or segment, only the *earliest* footprints are visible. When a rabbit enters a node that is already colored, it chooses the direction of its next move: while the segment in front of it is colored differently from the node it stands on, it rotates 90 degrees clockwise. Rotating takes no time. After all rotations, the rabbit moves forward again with the same speed, and so on.

When a rabbit reaches the border of the grid, it starts to run over it in clockwise direction until he gets jailed. A rabbit is jailed when it moves into a node containing a trap.

If a rabbit meets some nonempty rabbit hole on its way, he immediately warns hole inhabitants and moves on, while the hole inhabitants immediately hide in deep tunnels and disappear from the garden forever.

Pasha picks a paintball, shoots it at some rabbit hole, waits until all the rabbits are jailed, picks another paintball, shoots it at another nonempty hole, waits again and so on. Each time, Pasha chooses the rabbit hole randomly. The choice of each nonempty rabbit hole is equiprobable. Pasha stops when all the rabbit holes are empty. He wants to know the number L, the expected sum of lengths of rabbit paths.

You are to write a program that will find the desired value of L for the given grid with rabbit holes.

Input

The first line of input contains two positive integers N and M ($2 \le N, M \le 30$) — the number of rows and columns of the garden's grid respectively. The second line contains the only integer K — the number of rabbit holes in the garden ($1 \le K \le (N - 1) \cdot (M - 1)$). The following K lines contain two integers R_i and C_i each — row number and column number of i-th rabbit hole ($0 < R_i < N$, $0 < C_i < M$). Each node contains at most one rabbit hole.

Output

Output the expected number L with an absolute or relative error not exceeding 10^{-6}.

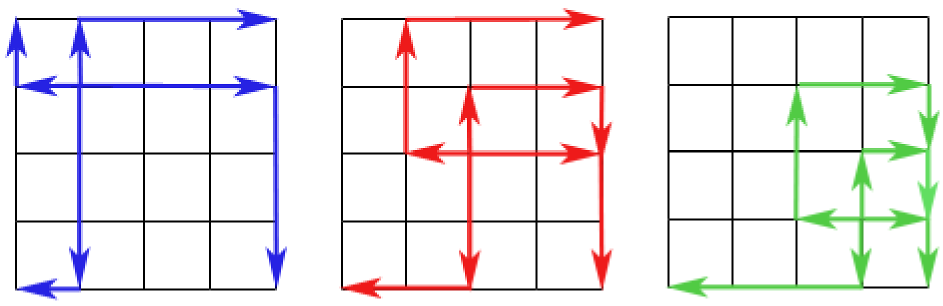
Example

standard input	standard output

4 4
3
2 2
1 1
3 3
	54.666667

Note

If Pasha shoots at the rabbit holes in the following order: $[1, 1]$, $[2, 2]$, $[3, 3]$, the routes for rabbits will be the following (in the same order):



Possible routes for rabbits in the sample

Problem K. Key Arrays

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

In the some database an array of integers A is used as key. During the database work, there are two possible types of modifications:

```
procedure change(A[], i, j, C) {
    for (k = i; k <= j; k++)
        A[k] += C;
}

procedure reverse(A[], i, j) {
    for (k = 0; k + i < j - k; k++)
    {
        t = A[i+k];
        A[i+k] = A[j-k];
        A[j-k] = t;
    }
}
```

Besides of modifications, three types of queries are possible:

```
function getSum(A[], i, j) {
    result = 0;
    for (k = i; k <= j; k++)
        result += A[k];
    return result;
}

function getMin(A[], i, j) {
    result = A[i];
    for (k = i; k <= j; k++)
        if (A[k] < result) result = A[k];
    return result;
}

function getMax(A[], i, j) {
    result = A[i];
    for (k = i; k <= j; k++)
        if (A[k] > result) result = A[k];
    return result;
}
```

Your work is to implement the processing of key arrays.

Input

The first line of the input file contains N — the number of elements in key array A ($1 \le N \le 10^5$). The second line contains N integer numbers — initial values of array A ($-10^6 \le a_i \le 10^6$). The third line contains M — the number of operations with the array. The next M lines contain descriptions of operations ($1 \le M \le 10^5$). Calling procedure `change` is described by `change` i j C ($1 \le i \le j \le N$, $-1000 \le C \le 1000$). Calling procedure `reverse` is described by `reverse` i j ($1 \le i \le j \le N$). Calling all three query functions is described by `query` i j ($1 \le i \le j \le N$).

Output

For every `query` operation write one line with three numbers — result of functions `getSum`, `getMin`, `getMax`.

Example

standard input	standard output

5
1 2 3 4 5
5
change 2 3 1
query 2 3
reverse 1 4
query 2 3
query 1 5
	7 3 4
	7 3 4
	17 1 5