Problem A. Create the Best Pet

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

Petya plays a multiplayer computer game. The hero which represents Petya in the game can have up to three pets.

Each pet has eight parameters responsible for its attack and defense power. Each parameter is an integer from 1 to 999. Numbers greater than 500 primarily increase attack power, and lesser numbers are good for defense power. Generally, the more each parameter differs from 500 to either side, the better. Furthermore, each pet has eight genetic parameters which define how it looks. Lastly, each pet has a name.

When a player decides to buy a pet in the game shop, only its name can be chosen. After that, an algorithm is run which picks power and genetic parameters randomly. Petya already bought one pet but was not satisfied: the power parameters turned out to be mediocre. He firmly decided to do whatever it takes to get a strong pet next time. But how to achieve it?

After some research, Petya got his hands on a piece of code which is responsible for pet generation, written in scripting language XyzzyLang. Here it is:

```
CreateRandomPet (Name):
    Seed := Now xor Hash (Name) xor Salt
    return Pet:
        Pet.Name := Name
        for I := 1, 2, ..., 8:
            Pet.Power[I] := GenerateRandomPower
        for J := 1, 2, ..., 8:
            Pet.Gene[J] := Random mod 5

Hash (String):
    Result := 0
    for I := 1, 2, ..., String.Length:
        Result := Result * 31 + String[I].AsciiCode
    return Result

Random:
    Seed := Seed * 1234567893 + 151515151
    return Seed

GenerateRandomPower:
    Result := 500
    for I := 1, 2, ..., 10000:
        Result := Result + Random mod 3 - 1
    return Result
```

Petya did not encounter this language before, but already learned that all numbers in the program are integers, and all calculations are performed modulo 2^{31}. As can be seen from the code, the pet's parameters are fully determined by three numbers: Now, Hash(Name), and Salt. Petya knows that the function Now returns the current moment in unix timestamp format (the number of seconds passed since January 1, 1970). The function Hash uses ASCII codes of characters (65–90 for `A`–`Z` and 97–122 for `a`–`z`). However, Petya didn't find the value of the number Salt anywhere.

According to Petya, the quality of a pet is the average deviation of its power parameters from the number 500: the sum of absolute differences $|500 - \texttt{Power[I]}|$ divided by 8. Note that this doesn't take genetic parameters into account.

Petya already came up with a good name for his next pet, and wants to create it at moment Start. However, in order to get a strong pet, he can wait from 0 to Wait seconds inclusive. Under these conditions, determine when exactly should Petya make a purchase in order to get a pet with the highest possible quality.

In all tests for this problem, the number Salt is equal to the same secret value from 0 to $2^{31} - 1$. The example shows the first pet that Petya bought.

Input

The first line contains two integers Start and Wait, followed by the word Name ($1\,535\,814\,000 \le$ Start $\le 2\,000\,000\,000$, $0 \le$ Wait ≤ 120). The word Name is from 1 to 20 characters long and can contain only lowercase and uppercase English letters.

Output

On the first line, print an integer: the moment to generate the best pet, and then a real number: the quality of this pet. The quality should be printed as precisely as possible. On the second line, print eight numbers: the power parameters of the generated pet in their respective order. On the third line, print eight numbers: the genetic parameters of the pet in their respective order.

If it is possible to generate several pets with the same maximum quality, print the one which can be generated earlier.

Example

standard input	standard output
1535814000 0 Murik	1535814000 69.250 484 467 469 363 621 504 291 503 1 3 2 3 0 0 4 2

Problem B. Graph and Machine

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	512 mebibytes

John's father recently passed away and left John a colored graph and a machine. The colored graph was simply a connected undirected graph with labels 0 or 1 on each of its vertices. The machine was something more peculiar.

A machine of order n is an acyclic directed graph with one source (a vertex with no incoming edges) and two sinks (vertices with no outgoing edges). One of the sinks is labeled with 0 and the other is labeled with 1. Each of the remaining vertices including the source is labeled with an integer from $\{1, \ldots, n\}$ and has exactly two outer edges: one labeled with 0 and the other labeled with 1. Also **on every path from the source to a sink, all labels of the non-sink vertices are distinct**.

A machine of order n computes a function from $\{0,1\}^n$ to $\{0,1\}$. Let us define it recursively. For 0-sink, the function is 0 on every input, for 1-sink, it is 1 for every input. For a non-sink vertex v labeled with i,

$$f_v(x_1, \ldots, x_n) = \begin{cases} f_{t_0}(x_1, \ldots, x_n) & \text{if } x_i = 0 \\ f_{t_1}(x_1, \ldots, x_n) & \text{if } x_i = 1 \end{cases}$$

where t_j is the end of the edge from v labeled with j for $j \in \{0,1\}$. The function calculated by a machine with the source s is f_s.

In his will, Jonh's father wrote that he had worked on the machine for years in order to calculate the edge-coloring function of the colored graph he had given to John. All he asks John is to check if the machine calculates this function correctly.

The edge-coloring function $\mathrm{EC}(x_1, \ldots, x_m)$ of a colored graph G with ℓ vertices and m edges with vertex-labels $c_1, \ldots, c_\ell$ is a function from $\{0,1\}^m$ to $\{0,1\}$. It equals 1 if and only if for every vertex v with incident edges $e_1, \ldots, e_k$, the following equality holds: $c_v = \bigoplus_{i=1}^{k} x_{e_i}$. In other words, the parity of the sum of values on edges incident to v is c_v.

You are asked to check if the given machine calculates the edge-coloring function of the given graph, and if it is not, find the coloring of edges x such that $\mathrm{EC}(x) \neq f(x)$, where f is the function calculated by the machine.

Input

The first line contains five integers N, m, s, t_0, and t_1: the number of nodes in the machine, the order of the machine, the index of the source and the indices of the 0-sink and 1-sink respectively ($1 \le s, t_0, t_1 \le N \le 300\,000$; $N \ge 3$; $1 \le m \le 300\,000$; $t_0 \neq t_1$; $s \neq t_0$; $s \neq t_1$). The i-th of the next N lines describes the i-th node of the machine. It contains three integers o_0, o_1 and ℓ: the index of the end node of the outer edge from the node i labeled with 0, this index for the edge labeled with 1, and the label of the node i itself ($-1 \le o_0, o_1 \le N$; $-1 \le \ell \le m$). If i is a sink, $o_0 = o_1 = \ell = -1$. The values o_0, o_1 and ℓ are never equal to zero.

It is guaranteed that

- the graph of the machine is acyclic;
- o_0, o_1 or ℓ are equal to -1 if and only if the node is a sink;
- on every path from the source to a sink, all labels of non-sink vertices are unique;
- all vertices except maybe one of the sinks are reachable from s.

The next line contains one integer k, the number of vertices in the colored graph G ($1 \le k \le 300\,000$). The number of edges in this graph is m. The following line contains k integers $c_1, c_2, \ldots, c_k$, the labels of the vertices of G (each c_i is either 0 or 1).

The last m lines contain descriptions of the edges of G. The i-th of these lines contains two integers a_i and b_i which describe an edge connecting a_i and b_i ($1 \le a_i, b_i \le k$; $a_i \ne b_i$). It is guaranteed that G is connected, but it may contain parallel edges.

Output

Print "YES" on the first line if the machine calculates the edge-coloring function correctly. Otherwise, print "NO" on the first line, and on the next line, print m characters $x_1, x_2, \ldots, x_m$ such that $\mathrm{EC}(x_1, x_2, \ldots, x_m) \ne f(x_1, x_2, \ldots, x_m)$, where f is the function computed by the machine. Each x_i must be either 0 or 1.

Examples

standard input	standard output
7 3 1 6 7 2 3 1 6 4 2 5 6 2 6 7 3 7 6 3 -1 -1 -1 -1 -1 -1 3 1 1 0 1 2 1 3 2 3	YES
7 3 1 6 7 2 3 1 6 4 2 5 6 2 6 7 3 6 7 3 -1 -1 -1 -1 -1 -1 3 1 1 0 1 2 1 3 2 3	NO 101
3 1 1 2 3 2 3 1 -1 -1 -1 -1 -1 -1 2 1 1 1 2	YES

Problem C. Clique Festival

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

John has a graph with n vertices labeled with integers $1, \ldots, n$. Initially, there are no edges in the graph. Then John modifies the graph k times, each time adding a clique to the graph. He chooses an integer a and a set S which is a non-empty subset of the set of integers $1, 2, \ldots, n$. For each unordered pair (i, j) such that $i, j \in S$ and $i \neq j$, John adds an undirected edge between the vertices i and j with weight a. It is possible that parallel edges appear in John's graph.

The distance between vertices u and v is defined as follows. Denote $d_{u,v}$ as the minimum weight of the edge between vertices u and v, or ∞ if there is no such edge. Then $\text{dist}(u, v) = \min\limits_{i_1,\ldots,i_p} \left(d_{u,i_1} + d_{i_1,i_2} + \ldots + d_{i_{p-1},i_p} + d_{i_p,v}\right)$. In other words, the distance is the length of the shortest path between u and v.

Your task is to calculate $\sum\limits_{i=1}^{n-1} \sum\limits_{j=i+1}^{n} dist(i, j)$. It is guaranteed that all summands are finite.

Input

The first line contains two integers n and k ($1 \le n \le 100\,000$, $1 \le k \le 18$). The next k lines contain the descriptions of the added cliques. Each of these lines contains integers a ($1 \le a \le 10^7$), $|S|$ ($1 \le |S| \le n$), and then $|S|$ integers $s_1, \ldots, s_{|S|}$ ($1 \le s_i \le n$, all s_i are distinct). These are the weight of edges in the clique, the number of vertices in the clique, and the labels of these vertices, respectively.

The sum of all $|S|$ in the input does not exceed 300 000.

Output

Output a single integer: the answer to the problem.

Examples

standard input	standard output
10 3 10 5 1 2 3 4 5 10 5 6 7 8 9 10 1 2 5 6	625
3 2 1 2 1 2 1 2 2 3	4

Problem D. Distance in Crosses

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

Consider a plane partitioned into squares with side 1. Let us choose a square and draw coordinate axes from its center parallel to its sides.

Next, let us draw a cross consisting of the central square and its four neighbors: the squares which share a side with it. Then pick the square centered at point $(2, 1)$ and draw another cross consisting of this square and its four neighbors. Tile the whole plane with such crosses: their centers will be in points with coordinates $(2i + j, i - 2j)$ for all possible integer i and j. The tiling is shown alongside the examples.

Emilia stands at the center of some square on the plane. In one step, she can move from a square to one of its neighbors. If a step brings her to a different cross of the tiling, she has to pay one coin for this step. Steps such that Emilia remains in the same cross are free.

Let the *distance in crosses* between two squares A and B be the minimum possible number of coins that Emilia has to pay in order to get from A to B. You are given the coordinates of two points on the plane: center of the initial square and center of the target square. Find the distance in crosses between them.

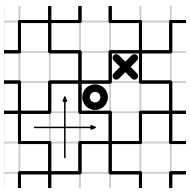
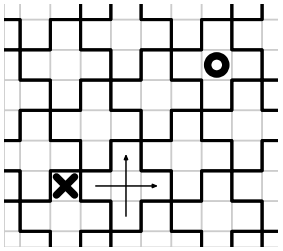
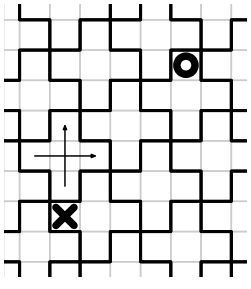
Input

The first line contains two integers x_1 and y_1, the coordinates of the initial square. The second line contains two integers x_2 and y_2, the coordinates of the destination square. All given coordinates do not exceed 10^9 by absolute value.

Output

Print one integer: the distance in crosses from the initial square to the destination square.

Examples

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

Problem E. Lui and Lines

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

Lui is a magician, and recently he mastered traveling in n-dimensional space. In order to adjust the settings of his new multidimensional spaceship, he has to find the distance between two lines in n-dimensional space. Each line is defined by two distinct points lying on it. The distance between lines is the smallest distance between a pair of points (x, y) such that x lies on the first line and y lies on the second one.

The distance between the points $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$ in n-dimensional space is defined as

$$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}.$$

The n-dimensional line containing two points a and b could be formally defined as the set of points $(ta_1 + (1-t)b_1, ta_2 + (1-t)b_2, \ldots, ta_n + (1-t)b_n)$ for all real values of t.

Input

The first line contains the integer n ($1 \le n \le 100\,000$), the number of dimensions. The next four lines contain the description of four points a, b, c, d in n-dimensional space. The points a and b lie on the first line, and the points c and d lie on the second one ($a \ne b$, $c \ne d$). Each of these four lines contains n integers. All the numbers in the input do not exceed 10^5 by absolute value.

Output

Let d be the distance between the lines. Print d^2 as a fraction x/y such that $x \ge 0$, $y > 0$, and the greatest common divisor of x and y is equal to 1.

Examples

standard input	standard output
2 0 0 1 0 0 1 1 1	1/1
2 0 0 1 1 1 0 2 1	1/2

Problem F. Dominating Subarray

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

An array $x_1, \ldots, x_\ell$ is a *subarray* of an array $y_1, \ldots, y_m$ if there exists an integer i from 1 to $m - \ell + 1$ such that the following equalities hold: $y_i = x_1$, $y_{i+1} = x_2$, ..., $y_{i+\ell-1} = x_\ell$.

A subarray $b_1, \ldots, b_k$ of a is *k-dominating* if for any subarray $c_1, \ldots, c_k$ of a, the following inequalities hold: $b_1 \geq c_1$, $b_2 \geq c_2$, ..., $b_k \geq c_k$.

You are given an array $a_1, \ldots, a_n$. Find any occurrence of a k-dominating subarray of a if it exists.

Input

The first line contains two integers n and k ($1 \leq k \leq n \leq 100\,000$), the length of the array a and the parameter. The second line contains n integers $a_1, \ldots, a_n$ ($1 \leq a_i \leq 10^6$), the elements of the array.

Output

Print "-1" if there is no k-dominating subarray in a. Otherwise, print an integer i from 1 to $n - k + 1$ such that $a_i, \ldots, a_{i+k-1}$ is a k-dominating subarray of a.

Examples

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

Problem G. Least Number

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

You are given a number N and a digit d. Consider all numbers which **do not contain** digit d in their decimal notation. Find the minimum such number for which the sum of its digits is exactly N.

Input

The first line contains two integers N and d ($2 \le N \le 10^6$, $0 \le d \le 9$).

Output

Print one integer: the minimum number which does not contain digit d and has the sum of its digits equal to N.

Example

standard input	standard output
14 9	68

Problem H. Galactic Governments

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

Currently, there are n galactic governments in the k-dimensional cubic universe. The universe is a cube with two opposite vertices $(0, \ldots, 0)$ and $(C, \ldots, C)$ and sides parallel to coordinate axes. Formally, the set of points in the universe is

$$U = \{(x_1, \ldots, x_k) \in \mathbb{R}^k : 0 \le x_i \le C\}.$$

Each galactic government claims that its territory is a parallelepiped with sides parallel to coordinate axes. The i-th government claims a parallelepiped with two opposite vertices $(a_{i,1}, \ldots, a_{i,k})$ and $(b_{i,1}, \ldots, b_{i,k})$ such that $a_{i,j} < b_{i,j}$ for all j. Formally, the i-th government claims the set of points

$$G_i = \{(x_1, \ldots, x_k) \in U : a_{i,j} \le x_j \le b_{i,j}\}.$$

Note that some pieces of territory can be claimed by multiple governments.

Rick tries to find a point which is not claimed by any of the galactic governments. He has noticed that $a_{i,j}$ is an integer for all i from 1 to n and all j from 1 to k. Rick knows that it implies that an unclaimed point exists if and only if there exists an unclaimed point $\left(\alpha_1 + \frac{1}{2}, \alpha_2 + \frac{1}{2}, \ldots, \alpha_k + \frac{1}{2}\right)$ where α_i *are all integers*. Rick likes integers, so he asks you to find $\alpha_1, \ldots, \alpha_k$ such that $\left(\alpha_1 + \frac{1}{2}, \alpha_2 + \frac{1}{2}, \ldots, \alpha_k + \frac{1}{2}\right)$ is a point in the universe and it does not belong in any of the $G_1, \ldots, G_n$. If there are multiple such points, Rick wants to find the lexicographically smallest one.

Point $\left(\beta_1 + \frac{1}{2}, \ldots, \beta_k + \frac{1}{2}\right)$ is lexicographically smaller than $\left(\gamma_1 + \frac{1}{2}, \ldots, \gamma_k + \frac{1}{2}\right)$ if there exists such j $(1 \le j \le k)$ such that for all $i < j$ we have $\beta_i = \gamma_i$, and $\beta_j < \gamma_j$.

Input

The first line contains three integers n, k, and C ($1 \le n \le 18$, $1 \le k \le 10$, $1 \le C \le 1000$). The i-th of the next n lines contains $2k$ integers: $a_{i,1}, \ldots, a_{i,k}$, $b_{i,1}, \ldots, b_{i,k}$ ($0 \le a_{i,j} < b_{i,j} \le C$ for every j from 1 to k).

Output

Print "NO" if all points in the universe are claimed by galactic governments. Otherwise, print "YES" on the first line, and on the second line, print k integers $\alpha_1, \ldots, \alpha_k$ such that $\left(\alpha_1 + \frac{1}{2}, \alpha_2 + \frac{1}{2}, \ldots, \alpha_k + \frac{1}{2}\right)$ is a point in the universe and it does not belong in any of the $G_1, \ldots, G_n$. If there are multiple solutions, print the lexicographically smallest one.

Examples

standard input	standard output
2 2 3 0 0 2 2 1 1 3 3	YES 0 2
1 3 5 0 0 0 5 5 5	NO

Problem I. Multiplication

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

This is an interactive problem.

Jury has chosen a secret odd number x between 1 and $2^{31} - 1$ inclusive. Your task is to guess it. In order to do that, jury gives you an even number n. Then you should output **exactly** n distinct integers between 0 and $2^{31} - 1$ inclusive. After that, jury will multiply each of these numbers by x and take the results modulo 2^{31}. Then jury will equiprobably choose some random subset of these new numbers of size $n/2$ and give this subset back to you in random order. After that, you should output the correct value of x.

In each test, x is chosen in advance and does not change.

Interaction Protocol

Initially, you are given one even integer n ($4 \le n \le 10^5$). After that, you should output n distinct integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 2^{31} - 1$) on a single line, separated by spaces. Next, you are given $n/2$ integers $b_1, b_2, \ldots, b_{n/2}$ ($0 \le b_i \le 2^{31} - 1$) created by the process described above, on a single line, separated by spaces. Finally, you should output a single odd integer x: the secret number chosen by the jury ($1 \le x \le 2^{31} - 1$).

To prevent output buffering, flush the output buffer after each printed line: this can be done by using, for example, `fflush (stdout)` in C or C++, `System.out.flush ()` in Java, `flush (output)` in Pascal, or `sys.stdout.flush ()` in Python. Also, do not forget to terminate each line of output with a newline character.

Example

standard input	standard output
4 9 6	 1 2 3 4 3

Problem J. Guess Two Strings

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

This is an interactive problem.

Jury has chosen two secret binary strings of length N and called them s and t such that s is not lexicographically greater than t. Your task is to find out which strings were chosen. To do that, you can ask the jury to generate up to Q strings. Each such string r will be generated in the following way:

1. start by assigning $r = s$ or $r = t$, choosing one of them randomly with equal probability,
2. randomly select K **distinct** positions in the string r so that each set of K positions has equal probability of being selected,
3. flip the digits at the selected positions in r: change all "0"s to "1"s and all "1"s to "0"s,
4. give the modified string r to you.

Note that s and t don't change during generation of string r.

Your task is to correctly guess s and t.

Interaction Protocol

Initially, you are given a single line with three integers N, K, and Q ($N = 100$, $K = 15$, $Q = 100$): the length of the strings s and t, the number of positions to choose for flipping, and the maximum number of strings which can be generated.

To request another generated string, print a line containing a single "?" to the standard output. After that, you will be given a line containing a string of N binary digits: the newly generated string r. You can make no more than Q such requests.

When you are ready to make a guess, print a line in the format "! s t" where s and t are two strings of N binary digits each. After that, terminate your program gracefully.

To prevent output buffering, flush the output buffer after each printed line: this can be done by using, for example, `fflush (stdout)` in C or C++, `System.out.flush ()` in Java, `flush (output)` in Pascal, or `sys.stdout.flush ()` in Python. Also, do not forget to terminate each line of output with a newline character.

Example

standard input	standard output
4 1 42 1010 1110 0110 0010 0000 0100 0011 0111	 ? ? ? ? ? ? ? ? ! 0010 0110

Example explanation

This example violates the constraints, and is given only to illustrate the process of interaction. All tests in the testing system will satisfy all the constraints from the statement.

Problem K. Beautiful Tables

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

Alice has a rectangular table consisting of $n \times m$ squares. Some squares are empty, and other are filled with integers.

Alice thinks that a table is *beautiful* if:

- for every square which has neighbors both left and right, the number in it is half of the sum of numbers in these neighbors,
- for every square which has neighbors both up and down, the number in it is half of the sum of numbers in these neighbors.

Alice want to check if she can put numbers (not necessary integers) in all empty squares to make her table beautiful. Also, if she can, she is interested if there is an unique way to do it.

Input

The first line contains two integers n and m ($1 \le n, m \le 10$). The next n lines describe of the table. Each of these lines contains m tokens separated by spaces. Each token is either "`?`" if the respective square is empty, or the respective number if it is filled.

All the given numbers are integers not greater than 100 by absolute value. However, there are no such constraints on the numbers which Alice can put in empty cells.

Output

If there is no way to make the table beautiful, print the only word "`None`".

If there is an unique way to make the table beautiful, print the word "`Unique`" on the first line, and then n lines containing the table after filling all empty squares.

If there is more than one way to make the table beautiful, print the word "`Multiple`" on the first line, and then two different solutions: n lines containing one example of the table, then a line with the word "`and`", and then n more lines containing a different example of the table. Two tables are considered different if there is at least one square in which they differ.

The numbers in tables should be printed as rational fractions with numerator no more than 10^{18} by absolute value and positive denominator no more than 10^{18}, separated by a single character "`/`". Numerator and denominator should be coprime, and a denominator equal to one can be omitted, but only together with the character "`/`". Please refer to the examples for more details.

Examples

standard input	standard output
3 5 1 2 3 ? ? ? 5 ? ? ? ? ? ? 0 ?	Unique 1/1 2/1 3/1 4/1 5/1 13/2 5/1 7/2 2/1 1/2 12/1 8/1 4/1 0/1 -4/1
3 3 1 2 3 7 ? 4 ? 6 5	None
2 2 1 2 ? 4	Multiple 1/1 2/1 3/1 4/1 and 1/1 2/1 2/1 4/1

Problem L. Three Balls

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

Do you like problems with short and clear statement? You are given three balls in three-dimensional space. The balls do not intersect and have different radii. Find the volume of their convex hull.

Input

The input consists of three lines, each line describes one ball.

Each line contains four integers x_k, y_k, z_k, r_k: the coordinates of the center of the k-th ball and its radius ($1 \le r_k \le 100$, $0 \le x_k, y_k, z_k \le 100$).

It is guaranteed that the balls do not intersect and do not touch. In addition, $r_1 < r_2 < r_3$.

Output

Print the volume of the convex hull. The output will be accepted if it has absolute or relative error at most 10^{-7}.

Example

standard input	standard output
10 0 0 1 0 10 0 2 0 0 10 3	691.832383333484585819