

Problem A. Combinatory logic

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

Combinatory logic is a field in mathematical logic developed in the first half of the *XX* century by Moses Sheinfinkel and Haskell Curry. Main elements of combinatory logic are constants, variables and terms. To construct terms we use the following induction rules:

1. if x is a variable then x is a combinatory term;
2. if c is a primitive function then c is a combinatory term;
3. if A and B are combinatory terms then (AB) is a combinatory term;
4. there are no other combinatory terms.

The expression (AB) means the operation of applying function (combinator) A to its argument B . To shorten the notations, the expression may have omitted brackets, if so they can be recovering by applying associativity rule:

$$ABC = ((AB)C).$$

So according the rule above, let's assume that the expression ABC is also the correct form of combinatory term.

Let's note variables as lowercase letters and combinators as uppercase letters. In the problem only the combinators can be constants.

Now let's define the simple combinators I , K , and S :

- $(Ia) = a$;
- $((Ka)b) = a$;
- $((Sb)c) = ac(bc)$.

If we don't go deep into details of combinatory logic, we may compute the combinatory terms only using the following simple rules:

1. find the leftmost combinator that can be computed;
2. omit all left brackets behind the combinator with its pairs if they are present;
3. apply the combinator by taking one or more next terms as parameters:
 - one term for combinator I
 $Ia = a$;
 - two terms for combinator K
 $Kab = a$;
 - three terms for combinator S
 $Sabc = ac(bc)$.
4. if there are not enough terms to compute the combinator the term remains the same.

Let's consider some more examples:

$$S(KII)ab = (KII)b(ab) = (I)b(ab) = Ib(ab) = b(ab);$$

$$SIISa = IS(IS)a = S(IS)a = S(S)a = SSa;$$

$$\begin{aligned} S(KK)I(Kab)b &= (KK)(Kab)(I(Kab))b = KK(Kab)(I(Kab))b = \\ &= K(I(Kab))b = (I(Kab)) = I(Kab) = (Kab) = (a). \end{aligned}$$

Given the combinator P written by using primitive combinators I , K , S and brackets. Compute the application of P to the sequence containing the first n lowercase Latin letters.

Input

The first line contains one integer number n ($1 \leq n \leq 8$). The second line contain the description of combinator P (the string of symbols “I”, “K”, “S”, “(” and “)”). P must be a combinatory term. The length of string is at least 1 and at most 150 000. The maximum nesting level of brackets does not exceed 100. The total number of applied combinators in computation does not exceed 50 000.

Output

Output the resulting string of application of P to the sequence of the first n Latin letters. It is guaranteed, that the string length will not exceed 8 and it will contain only lowercase Latin letters.

Examples

standard input	standard output
1 I	a
2 K	a
1 SII	aa
1 S(SII)I	aaa
3 S((S(KS)K)(S(KS)K)S)(KK)	acb
2 SS(K(SKK))	abb

Problem B. Sequences

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

Let's consider the sequences of n symbols. Let's say the sequence is k -simple if the symbols at positions i and $i + k$ ($1 \leq i \leq n - k$) are different. Determine the number of k -simple sequences consisting of the first m Latin letters.

Input

The first line has three integer numbers n ($1 \leq n \leq 40$), m ($1 \leq m \leq 8$) and k ($1 \leq k \leq 8$).

Output

Output one integer equal to the number of such sequences.

Example

standard input	standard output
3 2 2	4

Note

In the example the sequences are: aab, abb, baa, bba.

Problem C. Game with stones

Input file: `standard input`
Output file: `standard output`
Time limit: 1 second
Memory limit: 256 megabytes

Two players play the following game. There are n heaps of stones, where i -th heap has a_i stones. The players make moves in turn. On the each turn the active player chooses the heap and removes an arbitrary positive number of stones. The player removing the last stone will lose the game.

For the given starting position determine if the first player will win and if it is true find the winning move. Consider that two players play optimally.

Input

The first line has one integer n ($1 \leq n \leq 100$) denotes the number of heaps.

The second line has n integer numbers a_i ($1 \leq a_i \leq 100$) that is equal to number of stones in the i -th heap.

Output

Output “Lose” if the player who goes first lose the game and “Win” otherwise.

If the first players wins the second line should contain two integers equal to the index of the heap and the number of stones to be removed from the heap on the first turn, respectively.

Examples

standard input	standard output
2 3 3	Lose
2 3 2	Win 1 1
3 3 3 3	Win 1 3

Problem D. Equal Vertices

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

Compare all vertices of a finite vertex-labeled directed graph in which every vertex has at most one outgoing edge.

Two vertices are equal if they cannot be distinguished by the following process:

1. Compare labels. Whenever they differ, the process stops and reports vertices are not equal.
2. Compare outgoing degrees. Whenever they differ, the process stops and reports vertices are not equal.
3. When both vertices have no outgoing edges, the process stops and reports vertices are equal.
4. Follow outgoing edges and repeat the process.

Please note, when the process reports vertices are equal, it implies that vertices cannot be distinguished.

Input

The first line contains a single integer n ($2 \leq n \leq 10^5$): the number of vertices.

The following n lines describe vertices. Each description contains a lowercase English letter (vertex label) and number t separated by space. $t = 0$ means that vertex has no outgoing edges. Otherwise, it has an edge to t -th vertex. Vertices are numbered from 1 to n in the order they given.

Output

Output exactly n lines.

On the i -th line write the equality class number of the i -th vertex. Equal vertices should have the same class number.

Classes are numbered as follows: the first vertex belongs to the first class, the next vertex that doesn't belong to the first class belongs to the second class, and so on.

Example

standard input	standard output
4	1
a 2	2
b 3	3
b 0	2
b 3	

Problem E. Pinball

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

Pinball is an arcade game. The player gets the points by manipulating the metal ball in the field by flippers. The main goal of the game is to get the maximum possible number of points.

The field usually contains many different objects, but in the problem the field has use only one kind of objects: bumpers.

Geometrically the bumper is the disk of the certain size. If the ball hits the bumper, the player earns some points. Then the ball may change its direction and moves till it hits another bumper.

The ball can move only to a bumper that is visible from the previous one i.e. it is possible to draw the straight line between centers of the bumpers that doesn't intersect and doesn't touch other bumpers. Also the ball can not return to the bumper that it hit before going to the current one. In other words if ball hit bumper a then bumper b then it can not go back to a .

Let the center of the first bumper has the coordinates (x_1, y_1) and its radius is equal to r_1 , the center and radius of the second bumper are x_2, y_2 and r_2 correspondingly. Then the travel time between the bumpers is equal to $\lceil \frac{\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}-r_1-r_2}{20} \rceil$ seconds (rounded up to nearest integer).

The game starts at $t = 0$ where the ball hit the top bumper (the center of bumper has the maximum y coordinate; if there are several such bumpers we take the bumper with maximum y coordinate and maximum x coordinate), the player earns points for hitting the bumper and the direction of the ball hasn't been determined yet.

Write a program that will process two types of queries for the given field.

Input

The first line of the input file contains the integer number n ($3 \leq n \leq 7$) denotes the number of bumpers. Each of the following n lines contain 4 integers x_i, y_i, r_i and s_i ($0 \leq x_i, y_i \leq 50, 1 \leq r_i \leq 5, -100 \leq s_i \leq 100$) defining coordinates of i -th bumper, its radius and number of points that player gets when the ball hit that bumper. Bumpers don't intersect and don't touch each other.

The next line contains one number q ($1 \leq q \leq 10\,000$) equals to the number of queries. The next q lines contain the integer number defining type of the query and another integer t_j ($0 \leq t_j \leq 10^9$) for the query of the first type and two integers t_j, x_j ($0 \leq t_j \leq 10^9, 1 \leq x_j \leq n$) for the query of the second type.

Output

For each query of the first type output one integer equals to the maximum possible number of game points that can be achieved among all moments of the first t_j seconds. Formally, let $p(x)$ be the number of points at the moment x . Find y where $p(y)$ is maximal and $y \in [0, t_j]$.

For each query of the second type output one integer equals to the maximum number of game points at t_j second after the start of the game when the ball previously hit the bumper x_j . If the situation is impossible, output just one symbol "#".

All answers to the queries must be placed in different lines.

Example

standard input	standard output
3	1
0 0 1 1	1
50 0 1 1	2
0 50 1 1	2
20	2
1 1	3
1 2	3
1 3	3
1 4	3
1 5	4
1 6	#
1 7	#
1 8	2
1 9	2
1 10	2
2 1 1	#
2 2 1	3
2 3 1	3
2 4 1	3
2 5 1	#
2 6 1	
2 7 1	
2 8 1	
2 9 1	
2 10 1	

Problem F. Combinatory logic 2

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

Combinatory logic is a field in mathematical logic developed in the first half of the *XX* century by Moses Sheinfinkel and Haskell Curry. Main elements of combinatory logic are constants, variables and terms. To construct terms we use the following induction rules:

1. if x is a variable then x is a combinatory term;
2. if c is a primitive function then c is a combinatory term;
3. if A and B are combinatory terms then (AB) is a combinatory term;
4. there are no other combinatory terms.

The expression (AB) means the operation of applying function (combinator) A to its argument B . To shorten the notations, the expression may have omitted brackets, if so they can be recovering by applying associativity rule:

$$ABC = ((AB)C).$$

So according the rule above, let's assume that the expression ABC is also the correct form of combinatory term.

Let's note variables as lowercase letters and combinators as uppercase letters. In the problem only the combinators can be constants.

Now let's define the simple combinators I , K , and S :

- $(Ia) = a$;
- $((Ka)b) = a$;
- $((S a)b)c) = ac(bc)$.

If we don't go deep into details of combinatory logic, we may compute the combinatory terms only using the following simple rules:

1. find the leftmost combinator that can be computed;
2. omit all left brackets behind the combinator with its pairs if they are present;
3. apply the combinator by taking one or more next terms as parameters:
 - one term for combinator I
 $Ia = a$;
 - two terms for combinator K
 $Kab = a$;
 - three terms for combinator S
 $Sabc = ac(bc)$.
4. if there are not enough terms to compute the combinator the term remains the same.

Let's consider some more examples:

$$S(KII)ab = (KII)b(ab) = (I)b(ab) = Ib(ab) = b(ab);$$

$$SIISa = IS(IS)a = S(IS)a = S(S)a = SSa;$$

$$\begin{aligned} S(KK)I(Kab)b &= (KK)(Kab)(I(Kab))b = KK(Kab)(I(Kab))b = \\ &= K(I(Kab))b = (I(Kab)) = I(Kab) = (Kab) = (a). \end{aligned}$$

Let's assume there was a combinator P that was applied to the sequence of first n lowercase Latin letters and the result of the application is the sequence X . Given the number n and the sequence X , find any expression P made by primitive combinators I , K , S and brackets.

Input

The first line contains one integer number n ($1 \leq n \leq 8$). The second line contains a nonempty sequence up to 8 symbols. It can contain only the first n lowercase Latin letters.

Output

Output the string with no more than 400 000 symbols “I”, “K”, “S”, “(” and “)” describing combinatory term P . The maximum nesting level of brackets must not exceed 150 and the number of combinators in computation must not exceed 150 000. It is guaranteed that for every input at least one answer exists.

Examples

standard input	standard output
1 a	I
2 a	K
1 aa	SII
1 aaa	S(SII)I
3 acb	S((S(KS)K)(S(KS)K)S)(KK)
2 abb	SS(K(SKK))

Problem G. Minimum volume tetrahedron

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

There is the angle $OABC$ in the 3D Cartesian space with the vertex $O(0, 0, 0)$ and points $A(a_1, a_2, a_3)$, $B(b_1, b_2, b_3)$, $C(c_1, c_2, c_3)$ on the edges.

Point $P(p_1, p_2, p_3)$ is placed strictly inside angle $OABC$ (i.e. the vector OP has positive elements in basis OA, OB, OC). Draw the plane through point P to make the tetrahedron with the minimum possible volume formed by the plane and the angle.

Input

Four lines contains the coordinates of points A, B, C, P separated by spaces. Coordinates are positive integer numbers that don't exceed 10^3 . Vectors OA, OB and OC are linearly independent.

Output

Output the minimum volume of tetrahedron cut by plane from the given angle with absolute or relative precision 10^{-6} .

Example

standard input	standard output
1 2 3 2 3 1 2 5 3 2 4 3	2.53125

Problem H. Inscribed triangle 3

Input file: `standard input`
Output file: `standard output`
Time limit: 1 second
Memory limit: 256 megabytes

Lets take an arbitrary nondegenerate triangle and lets draw the closed polyline with three segments with the following rules:

1. each side of the triangle must have at least one vertex of the polyline;
2. each vertex of the polyline must lie on some triangle's side;
3. the total length of all segments of the polyline must be minimum possible;
4. if there are several such polylines, one with minimal product of lengths of its segments must be chosen.

Determine, if there exists a triangle whose lengths of segments of the polyline can be equal to a, b, c ?

Input

The first line of the input file contains three integer numbers a, b, c ($-1\,000 \leq a, b, c \leq 1\,000$).

Output

Output "Yes" if the lengths of segments of the polyline can be equal to a, b, c or "No" otherwise.

Examples

standard input	standard output
2 3 4	Yes
-1 -1 -1	No

Problem I. $A^2 + \dots + B^2$

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

$$N = \sum_{k=A}^B k^2$$

Input

The only line contains two integer numbers A and B ($-10^{10^6} \leq A \leq B \leq 10^{10^6}$), separated by a space.

Output

Output single line containing digital root¹ of the number N .

Example

standard input	standard output
1 2	5

¹The digital root of a number is the number obtained by adding all the digits, then adding the digits of that number, and then continuing until a single-digit number is reached.

Problem J. Birthdays

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

Misha wants to get a job in the Kolya's department. He doesn't know the number of employees in the firm and assumes that there could be any number of employees with equal probability of cases.

To verify Misha mathematical abilities Kolya asked him to determine the number of employees knowing only the number of employees with the same birthdays.

All employees can be separated in groups, where all employees have the same birthday. The birthdays of two employees of the different group are different.

The all employees can be separated to the following groups. There are m_2 groups of two employees, m_3 groups of 3 employees, ..., m_s groups of s employees and other groups that contain only one employee.

Assuming that there are d days in any year and probability to have birthday is equal for any day in the year, help Misha find the most probable number of employees in the firm.

Input

The first line contains two integers s and d ($365 \leq d \leq 40\,000$, $2 \leq s \leq d$). The second line contains $s - 1$ integers $m_2, m_3, \dots, m_s$ ($0 \leq \sum_{i=2}^s m_i \leq d$).

Output

Output one positive integer number n denotes the most probable number of employees. If there are several possible answers output the maximum one.

Examples

standard input	standard output
2 365 1	28
4 365 3 5 2	113

