Problem A. Circular Roads

Input file:	`circular.in`
Output file:	`circular.out`
Time limit:	6 seconds
Memory limit:	256 megabytes

The government of St. Andyburg once decided to build a circular road around the city. However, the funds were cut, so only the part of the road was built. A couple of years later the decision to build the road was restored, but the new plan of the road was proposed. The construction of the new road had started, but the funds were cut again, and the road was not completed either.

Now the new government is investigating the new idea to build the road. Before making plans, they would like to investigate the situation with the two parts of the former roads.

Each road part is an arc of some circle. The first metrics the government would like to know is the shortest distance between the parts of the roads.

Input

The input file contains two lines, each line describes one road part. Each road part is described with five integer numbers: x, y, r, b and e — the coordinates of the center and the radius of the planned circular road, the starting angle and the ending angle of the built part. The angles are given in degrees, counterclockwise from the eastern direction (Oy axis looks northwards, Ox axis — eastwards), if $e < b$, the part of the road goes through eastern direction. All coordinates do not exceed 1000 by their absolute values, radii are positive and do not exceed 1000, b and e are from 0 to 359, inclusive, $b \neq e$.

Output

Output one real number — the shortest distance between road parts. Your answer must be accurate up to 10^{-5}.

Example

circular.in	circular.out
0 0 4 270 90 6 0 1 90 240	1.0
0 0 4 270 90 5 0 2 90 240	0.0

Problem B. Coffin Factory

Input file:	`coffin.in`
Output file:	`coffin.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Jerry works at a coffin factory. The factory produces two types of coffins: simple plywood coffins for poor people, and nice solid oak coffins for rich people.

The production of a coffin consists of two steps: knocking the coffin together out of planks, and polishing the resulting coffin. The steps are performed by two different teams, each team works in its own workshop. Between the steps the coffin must be transported from one workshop to another. This is exactly Jerry's job.

Both knocking the coffin together and polishing it takes exactly t minutes for both types of coffins. But the time Jerry needs to carry the coffin from one workshop to another depends on the type of the coffin. The plywood coffin is light and it takes c_1 minutes for Jerry to carry it from one workshop to another. But the oak coffin is heavy, so Jerry needs c_2 minutes. Without the coffin Jerry moves so fast that this time can be ignored.

The factory has received the order for a_1 plywood coffins and a_2 oak coffins. The director of the factory would like to minimize the total production time of the coffins — the time from the beginning of knocking together the first coffin till the moment the polishing of the last coffin is finished.

Help director to find the time needed to produce the coffins if they are produced in the optimal order.

Input

The input file contains five integer numbers: t, c_1, c_2, a_1, a_2 ($1 \le t, c_1, c_2 \le 10^4$, $c_1 \le c_2$, $0 \le a_1, a_2 \le 10^5$).

Output

Output one integer number — the minimal time needed to produce the required coffins.

Example

coffin.in	coffin.out
3 2 4 2 2	19

The factory must produce coffins in the following order in this case: oak, plywood, plywood, oak.

Problem C. Colorful Cycle Decomposition

Input file:	`colorful.in`
Output file:	`colorful.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Consider an undirected bipartite graph G with n vertices in each of its parts. Let each edge be colored in one of k possible colors.

The cycle in G is called *colorful* if no two adjacent edges in it have the same color. The collection of vertex disjoint cycles is called *colorful cycle decomposition* if all cycles in it are colorful, and each vertex of the graph belongs to exactly one of the cycles from the collection.

Given G, find its colorful cycle decomposition, or find out that it has none.

Input

The first line of the input file contains three integer numbers: n, m and k — the number of vertices in each part of the graph, the number of edges in the graph, and the number of colors used ($2 \le n \le 50$, $2 \le k \le 5$). Let vertices in the first part be numbered from 1 to n, and vertices in the second part be numbered from -1 to $-n$.

The following m lines contain three integer numbers each and describe edges. Each edge is described with the number of the vertex in the first part, the number of the vertex in the second part and the color of the edge. There are no parallel edges.

Output

If the colorful cycle decomposition exists, output "`YES`" at the first line of the output file. The second line must contain l — the number of cycles in the decomposition. The following l lines must describe cycles. Each cycle is described by a — the number of vertices in it, followed by a numbers — the vertices of the cycle, listed along any of its traversals.

If the colorful cycle decomposition doesn't exist, output "`NO`" at the first line of the output file.

Example

colorful.in	colorful.out
3 7 3 1 -1 1 1 -2 2 2 -1 3 2 -2 1 2 -3 2 3 -2 3 3 -3 1	YES 1 6 1 -1 2 -3 3 -2

Problem D. Delta, Kappa, Lambda

Input file:	`dkl.in`
Output file:	`dkl.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Let us consider a connected undirected graph G.

The degree of vertex v, denoted as $\deg v$, is the number of edges incident to it. The minimal degree of a vertex in G is denoted as $\delta(G)$.

The vertex cut size of G, denoted as $\kappa(G)$, is the minimal number of vertices that must be removed from the graph so that it contained at least two connected components. For the complete graph K_n let $\kappa(K_n) = n - 1$.

The edge cut size of G, denoted as $\lambda(G)$, is the minimal number of edges that must be removed from the graph so that it contained at least two connected components. For the singular graph K_1 that contains one vertex and no edges let $\lambda(K_1) = 0$.

You are given δ, κ and λ. Find the graph such that $\delta(G) = \delta$, $\kappa(G) = \kappa$ and $\lambda(G) = \lambda$, or determine that no such graph exists.

Input

The input file contains three integer numbers: δ, κ and λ ($1 \le \delta, \kappa, \lambda \le 10$).

Output

The first line of the output file must contain two integer numbers: n and m — the number of vertices and edges in the graph, respectively. Let the vertices of the graph be numbered from 1 to n. The following m lines must describe edges, each line must contain two integer numbers — the vertices connected by the corresponding edge. The graph must contain neither loops, nor parallel edges. The graph must contain at most 50 vertices.

If no such graph exists, output two zeroes at the first line of the output file.

Example

dkl.in	dkl.out
2 2 2	3 3 1 2 2 3 1 3
1 1 2	0 0

Problem E. Travelling Fly

Input file:	`fly.in`
Output file:	`fly.out`
Time limit:	10 seconds
Memory limit:	256 megabytes

Parents have bought a set of n cubes to little Johnny for his birthday. Johnny has glued the cubes together to form a small figure.

A fly is standing at a vertex of one of the cubes the figure is made of. It would like to travel to another vertex. The fly is very lazy, so it wouldn't like to fly, it would like to walk along the surface of the figure.

Help the fly to find the length of the shortest path from the vertex it is standing at to the vertex it would like to walk to.

Input

The first line of the input file contains n — the number of cubes ($1 \le n \le 15$). The following n lines describe cubes. Each cube is described by the coordinates of one of its vertices. From among the vertices the one with smaller coordinates is chosen. Coordinates are integer and do not exceed 15 by their absolute values. All cubes have edge length equal to 1 and are positioned in such a way that their edges are parallel to coordinate axes.

The next line contains three integer numbers — the coordinates of the point where the fly is standing. The next line contains three integer numbers — the coordinates of the point where the fly would like to walk to.

It is guaranteed that both starting and finishing points are at some vertex of a cube, and each of them is on the surface of the figure. The figure is connected in a sense that two cubes are called neighbors if they have a common face, and any two cubes are connected by a chain of neighbor cubes. There are no singularities, i.e. for each point and its small enough neighborhood both figure part that belongs to it and its complement are connected.

The fly can walk along the surface of the figure, that is — at the points that have a point not occupied by a cube in any neighborhood.

Output

Output one floating point number — the length of the shortest path from the starting point to the finishing point along the surface of the figure. You answer must be accurate up to 10^{-7}.

Example

fly.in	fly.out
4 0 0 0 0 1 0 1 0 0 1 1 0 1 1 0 1 1 1	3
1 0 0 0 0 0 0 1 1 1	2.2360679774997897

Problem F. Locally Linked Sequences

Input file:	`linked.in`
Output file:	`linked.out`
Time limit:	3 seconds
Memory limit:	256 megabytes

The sequence $a_1, a_2, \ldots, a_m$ is called *linked* if there are indices $i < j < k < l$ such that $a_i = a_k$, $a_j = a_l$, $a_i \neq a_j$.

The sequence $a_1, a_2, \ldots, a_n$ is called m*-locally linked* if for each i from 1 to $n - m + 1$ its subsegment $a_i, a_{i+1}, \ldots, a_{i+m-1}$ is linked.

You are given m, n and r. Find the number of m-locally linked sequences of length n, such that each of its elements is from the set of r elements. Print the answer modulo $10^9 + 7$.

Input

Input file contains three integer numbers: m, n and r ($4 \leq m \leq 8$, $m \leq n \leq 50$, $2 \leq r \leq 50$).

Output

Output one number — the number of locally linked sequences.

Example

linked.in	linked.out
4 5 3	6

The following sequences of length 5 are 4-locally linked: $(1, 2, 1, 2, 1)$, $(1, 3, 1, 3, 1)$, $(2, 1, 2, 1, 2)$, $(2, 3, 2, 3, 2)$, $(3, 1, 3, 1, 3)$, and $(3, 2, 3, 2, 3)$.

Problem G. Magicland Lottery

Input file:	`lottery.in`
Output file:	`lottery.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

The government of Magicland needs to get more money for its stabilizing fund. Therefore it decided to run a lottery among its citizens to gather some money.

The rules of the lottery are fairly simple. The special lokh... lottotron is loaded with balls of m different colors. There are c_1 balls of the first color, c_2 balls of the second color, etc..., c_m balls of the m-th color.

The balls are shaken well, and after that n balls are drawn randomly out of the lottotron, one after another. They are arranged in order they are drawn. These balls form the winning combination. Those who have guessed the winning combination will be awarded with a prize sum.

To determine the prize, the government needs to know the number of different possible combinations. Help them to calculate it.

Input

The first line of the input file contains m and n ($1 \le m \le 50$, $1 \le n \le 50$). The second line contains m integer numbers: $c_1, c_2, \ldots, c_m$ ($1 \le c_i \le 50$). The sum of all c_i is greater than or equal to n.

Output

Output the number of different possible combinations.

Example

lottery.in	lottery.out
2 2 1 2	3

There are three possible combinations: $(1, 2)$, $(2, 1)$ and $(2, 2)$.

Problem H. Move to Front

Input file:	`movetofront.in`
Output file:	`movetofront.out`
Time limit:	4 seconds
Memory limit:	256 megabytes

Corporal Studip likes to give orders to his squad. His favorite order is "move to front". He lines the squad up in a line and gives a series of orders. Each order is: "Soldiers from l_i to r_i — move to front!"

Let us number the soldiers in the initial line up from 1 to n, from left to right. The order "Soldiers from l_i to r_i — move to front!" makes the soldiers that are standing at the positions from l_i to r_i inclusive move to the beginning of the line, preserving their order.

For example, if at some moment the soldiers are standing in the following order: $2, 3, 6, 1, 5, 4$, after the order: "Soldiers from 2 to 4 — move to front!" the order of soldiers is $3, 6, 1, 2, 5, 4$. If, for example, the order "Soldiers from 3 to 4 — move to front!" follows, the new order of soldiers is $1, 2, 3, 6, 5, 4$.

Given the sequence of orders of corporal, find the final line up of the soldiers.

Input

The first line of the input file contains two integer numbers n and m ($2 \le n \le 100\,000$, $1 \le m \le 100\,000$) — the number of soldiers and the number of orders. The following m lines contain orders, each line contains two integer numbers l_i and r_i ($1 \le l_i \le r_i \le n$).

Output

Output n integer numbers — the order of soldiers in the final line up, after executing all orders.

Example

movetofront.in	movetofront.out
6 3 2 4 3 5 2 2	1 4 5 2 3 6

Problem I. Black and White Segments

Input file:	`segments.in`
Output file:	`segments.out`
Time limit:	3 seconds
Memory limit:	256 megabytes

There are several alternating black and white segments of integer length, located next to each other along the line.

Each second the longest segment is chosen (if there are several longest segments, the leftmost is chosen) and is changed using the following rules:

- If the segment chosen is white, it's split into two parts, the right part is colored black, and the left part is kept white.
- If the segment chosen is black, it's split into two parts, the left part is colored white, and the right part is kept black.

After that if there are two adjacent segments of equal color, they are joined together to form a single segment.

When splitting a segment of even length into two parts, the resulting parts are made equal. When splitting a segment of odd length into two parts, the resulting parts are made as close as possible, the part which preserves the color of the original segment is made 1 unit longer.

The process continues until one of the following is true: all segments have length 1, or some configuration of segments is repeated twice.

You are given the initial configuration of segments and the number t. You must find the number of seconds the process will continue, or detect that the process doesn't terminate within t seconds.

Input

The first line of the input file contains two integer numbers: n and t ($1 \le n \le 100\,000$, $1 \le t \le 100\,000$) — the number of segments in the initial configuration and the number of steps to simulate.

The second line contains a word "`white`" if the leftmost segment in the initial configuration is white, or a word "`black`" if the leftmost segment in the initial configuration is black.

The third line contains n integer numbers — the lengths of segments in the initial configuration, from left to right, in order they are located along the line. The length of each segment doesn't exceed 10^9.

Output

Output the number of seconds the process will continue, or t if the process doesn't terminate within t seconds.

Example

segments.in	segments.out
2 4 white 5 4	3
2 10 white 1000000000 1	10

Problem J. Separable Divisions

Input file:	`separable.in`
Output file:	`separable.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Consider a rectangle that consists of $m \times n$ unit squares. We can divide it into two parts by coloring some squares white and other squares black, so that both white and black parts are connected (the set of squares is called connected if one can walk from any square in the set to any other by stepping from a square to a square that shares an edge with it).

Let us call such division *separable* if one can move white and black parts as far apart from each other as desired, by continuously moving them without overlapping. For example, the division on the figure (a) is separable, but the division on the figure (b) is not.

(a) Separable division. (b) Nonseparable division.

Find the number of separable divisions of the rectangle.

Input

Input file contains m and n $(1 \le m, n \le 50)$.

Output

Output one number — the number of separable divisions of an $m \times n$ rectangle.

Example

separable.in	separable.out
4 4	470