Problem B. Connected Spanning Subgraph

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

Bobo has a connected undirected graph G with n vertices and m edges where vertices are conveniently labeled with $1, 2, \ldots, n$.

Bobo chooses a non-empty subset of edges such that the graph with the chosen edges is still connected. He would like to know the number of such subsets modulo 2.

Note that a graph is connected if, for any two vertices a and b, there exists a path which connects a and b.

Input

The input contains zero or more test cases, and is terminated by end-of-file. For each test case:

The first line contains two integers n and m ($2 \le n \le 2 \cdot 10^5$, $1 \le m \le 2 \cdot 10^5$).

The i-th of the following m lines contains two integers a_i and b_i which denote an edge between vertices a_i and b_i.

It is guaranteed that the sum of all m does not exceed $2 \cdot 10^5$, and all the given graphs are connected.

Output

For each test case, output an integer which denotes the remainder modulo 2.

Example

standard input	standard output

2 1
1 2
3 2
1 2
2 3
3 3
1 2
2 3
3 1
	1
	1
	0

Problem E. Maximum Flow

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

Bobo has an undirected graph with $(2n + 2)$ vertices conveniently labeled with the following pairs of integers: $(0, 0), (0, 1), \dots, (0, n), (1, 0), (1, 1), \dots, (1, n)$. The graph has three classes of edges.

- The edges of the first class connect vertices $(0, i-1)$ and $(0, i)$ with capacity a_i for $i \in \{1, 2, \dots, n\}$.
- The edges of the second class connect vertices $(1, i-1)$ and $(1, i)$ with capacity b_i for $i \in \{1, 2, \dots, n\}$.
- The edges of the third class connect vertices $(0, \lfloor \frac{i-1}{2} \rfloor)$ and $(1, \lfloor \frac{i}{2} \rfloor)$ with capacity c_i for $i \in \{1, 2, \dots, 2n + 1\}$.

Bobo would like to find the maximum flow from vertex $(0, 0)$ to vertex $(1, n)$.

Input

The input contains zero or more test cases, and is terminated by end-of-file. For each test case:

The first line contains an integer n ($1 \le n \le 5 \cdot 10^5$).

The second line contains n integers $a_1, a_2, \dots, a_n$.

The third line contains n integers $b_1, b_2, \dots, b_n$.

The fourth line contains $(2n + 1)$ integers $c_1, c_2, \dots, c_{2n+1}$.

The constraints are: $1 \le a_i, b_i, c_i \le 10^9$.

It is guaranteed that the number of test cases does not exceed 10^5, and the sum of all n does not exceed $5 \cdot 10^5$.

Output

For each test case, output an integer which denotes the maximum flow.

Example

standard input	standard output

1
2
2
1 3 1
3
1 4 7
2 5 8
2 3 3 2 1 2 4
	5
	6

Problem F. Rectangles Inside Rectangle

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

Bobo has a large rectangle with lower left and upper right corners at $(0, 0)$ and $(w, 10^6)$. He also has n small axis-parallel rectangles inside the large rectangle. The weight of the i-th rectangle is v_i. For each rectangle, either its left border or its right border (but not both) coincides with the left or right side of the large rectangle.

Bobo would like to choose a subset of small rectangles in such a manner that the rectangles may touch each other, but they do not overlap (that is, there are no points that belong to the interior of more than one rectangle). Among all the possibilities, he wants the one with the maximum possible sum of weights.

Input

The input contains zero or more test cases, and is terminated by end-of-file. For each test case:

The first line contains two integers n and w ($1 \le n \le 2000$, $2 \le w \le 10^6$) denoting the number of small rectangles and the width of the large rectangle.

The i-th of the following n lines contains five integers $type_i$, l_i, a_i, b_i and v_i ($type_i \in \{0, 1\}$, $0 \le a_i < b_i \le 10^6$, $1 \le l_i < w$, $0 \le v_i \le 10^6$) where v_i is the weight of the i-th rectangle. Here, $type_i = 0$ means the lower left and upper right corner of the i-th rectangle are $(0, a_i)$ and (l_i, b_i), while $type_i = 1$ means the lower left and upper right corner of the i-th rectangle are $(w - l_i, a_i)$ and (w, b_i).

It is guaranteed that for all $1 \le i < j \le n$, $a_i \ne a_j$, $a_i \ne b_j$, $b_i \ne a_j$ and $b_i \ne b_j$. Additionally, the sum of all n does not exceed 2000.

Output

For each test case, output an integer which denotes the maximum sum of weights.

Example

standard input	standard output

3 10
0 3 1 6 12
0 3 3 4 100
1 9 2 5 11
3 10
0 3 1 6 12
0 1 3 4 5
1 9 2 5 11
6 5
1 1 17 32 4
0 3 1 18 7
1 3 4 8 12
1 2 15 20 14
1 1 30 33 16
1 4 2 16 13
	100
	16
	42

Note

For the third test, Bobo can choose the 3-rd, 4-th and 5-th rectangles.

Problem G. Cute Panda

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

There are n pandas numbered from 1 to n, i-th of them has a_i donuts. There are also n bins numbered from 1 to n, i-th of them can hold b_i donuts. For any i from 1 to n, i-th panda can distribute his donuts to i-th and $(i \bmod n + 1)$-th bin.

Can you find a way to maximize the number of distributed donuts?

Input

The input contains zero or more test cases, and is terminated by end-of-file. For each test case:

The first line contains an integer n ($3 \le n \le 10^6$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$).

The third line contains n integers $b_1, b_2, \ldots, b_n$ ($0 \le b_i \le 10^9$).

It is guaranteed that the sum of all n does not exceed 10^6.

Output

For each test case, output an integer which denotes the maximum number of distributed donuts.

Example

standard input	standard output

5
8 4 8 3 10
1 0 4 5 1
5
9 4 10 0 4
3 5 2 2 1
	11
	13

Problem H. Order-Preserving Partition

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

Bobo has two permutations: $P = \{p_1, p_2, \ldots, p_n\}$ and $Q = \{q_1, q_2, q_3, q_4\}$. He would like to partition P into four non-empty and contiguous parts in such a manner that:

- The numbers in each part can be rearranged to form an *interval* of values: an increasing sequence where each element is greater than the previous by exactly one.
- For all $1 \le i < j \le 4$, $(s_i - s_j) \cdot (q_i - q_j) > 0$ where s_i is the minimum value in the i-th part.

Bobo wants to know the number of such partitions. As the number may be very large, you just need to print the answer modulo $(10^9 + 7)$.

Input

The input contains zero or more test cases, and is terminated by end-of-file. For each test case:

The first line contains an integer n, the length of the first permutation $(4 \le n \le 10^6)$.

The second line contains n integers $p_1, p_2, \ldots, p_n$.

The third line contains four integers q_1, q_2, q_3, q_4.

It is guaranteed that the sum of all n does not exceed 10^6.

Output

For each test case, output an integer denoting the answer.

Example

standard input	standard output

10
2 1 4 3 10 9 8 7 5 6
2 4 1 3
10
1 2 3 4 5 6 7 8 9 10
1 2 3 4
	0
	84

Problem J. Hamiltonian k-vertex-connected Graph

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

A graph (other than a complete graph) has connectivity k if k is the size of the smallest subset of vertices such that the graph becomes disconnected if you delete them.

A connected undirected graph G is called Hamiltonian if it has a Hamiltonian cycle: a cycle that visits each vertex exactly once (except for the vertex that is both the start and the end, which is visited twice).

Bobo would like to construct a Hamiltonian graph with n vertices which has connectivity k. Also, the number of edges in the graph should be minimum possible.

Input

The first line contains two integers n and k where n is the number of vertices in the graph ($3 \le n \le 100$, $1 \le k \le n - 2$).

Output

If there is no such graph, output -1 on a single line. Otherwise, output an integer m denoting the minimum number of edges. Then in each of the next m lines, output two integers x and y ($1 \le x, y \le n$, $x \ne y$) denoting an edge in the graph. In the following line, output a permutation of integers $1, 2, \ldots, n$ denoting a Hamiltonian cycle in the graph.

Example

standard input	standard output

`4 2`	`4`
	`1 2`
	`2 3`
	`3 4`
	`4 1`
	`1 2 3 4`

Problem K. Verifying A+B

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

Given a sum in form $a + b = c$, your job is to determine, if the string, containing it, does not have typos and that sum is correct

Input

The first and the only line of input contains a string, consisting of five space-separated elements: integer a, operation sign s_1, integer b, equation sign s_2 and integer c. It is guaranteed that integers are between 1 and 100, inclusive and both signs have ASCII codes between 33 and 126, inclusive.

Output

If operation sign equals to '`+`', equation sign equals to '`=`' and $a + b = c$, print '`YES`'.

Otherwise print '`NO`'.

Example

standard input	standard output
`2 + 2 = 4`	`YES`
`2 * 2 = 4`	`NO`
`4 = 2 + 2`	`NO`

Problem L. Make A Square

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

You want to make a square out of three colored rectangles of glass. You are given the sizes of the three panes of glass. Can you arrange them into a square? You may rotate the panes, but the panes may not overlap.

Input

The input consists of three lines. Each of the three lines gives the length and height of a pane of glass. It is guaranteed that the lengths and heights are integers between 1 and 100, inclusive.

Output

Print, on a single line, "`YES`", if the panes of glass can be combined to form a square; otherwise, print "`NO`".

Example

standard input	standard output

9 2
1 7
8 7
	YES

Problem M. Two Strings

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

Given two strings, each consisting of the lowercase English letters, you are to compare their *weight*.

String s_1 is considered *heavier*, than string s_2, if s_1 has more occurences of letter ‘z’ than s_2 does. If they have the same number of ‘z’, string s_1 is *heavier*, if it have the same number of ‘y’ etc. If those two strings are the anagrams, then their weight is considered equal.

Given two strings, calculate which one is heavier.

Input

First line of the input consists one integer n — number of test cases. Each test case consisting of two lines. First line contains string s_1, second line contains string s_2. Strings are non-empty; length of each string does not exceed 70.

Output

For each test case, if first string is heavier, than second, print 1, if their weight is equal,print 0, if second string is heavier, than first, print -1.

Example

standard input	standard output
3 zyzz axbycz ay by zxzx xzxz	1 -1 0

Problem N. Count them!

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

Given a list of points by their coordinated, determine how many triangles (with all angles strictly less than 180°) can be formed with those points.

Input

The first line of the input will contain an integer n ($1 \le n \le 50$), denoting the number of test cases in the file.

The first line of each test case contains one integer, k ($1 \le k \le 100$), denoting the number of points for that test case. The second line of each test case contains k ordered pairs of integers (separated by spaces) denoting the x and y coordinates of each point, respectively.

It is guaranteed that $-100 \le x \le 100$ and $-100 \le y \le 100$ for all coordinates. It is also guaranteed that each point given will be unique.

Output

For each test case, print one integer: number of formed triangles.

Example

standard input	standard output

2
4
1 1 2 2 3 4 4 4
3
2 4 6 8 7 9
	3
	0