Problem A. Fire On Field

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

We define A as a sequence of positive integers like the following.

Let $A[0] = 1$, $A[1] = 1$. For a positive integer $i \geq 2$, $A[i]$ is the smallest positive integer under the condition that no three terms $A[i-2k]$, $A[i-k]$, and $A[i]$ form an arithmetic progression for any integer $k > 0$ such that $i - 2k \geq 0$, that is, $A[i] - A[i-k] \neq A[i-k] - A[i-2k]$.

The sequence is uniquely determined like the following sequence: $A[0] = 1$, $A[1] = 1$, $A[2] = 2$, $A[3] = 1$, $A[4] = 1$, $A[5] = 2$, $A[6] = 2$, $A[7] = 4$, $A[8] = 4$, etc. The sequence element $A[2]$ cannot be 1 since otherwise $A[0] = 1$, $A[1] = 1$, $A[2] = 1$ form an arithmetic progression; here $i = 2$ and $k = 1$. If $A[2]$ is any integer larger than one, then the condition is satisfied. Therefore, $A[2]$ should be 2 which is the smallest positive integer among the possible ones. Similarly, it is easy to know that $A[3] = 1$. The sequence element $A[4]$ cannot be 3 since otherwise $A[4] - A[4-2] = A[4-2] - A[4-2 \times 2]$; here $i = 4$ and $k = 2$. Other natural numbers like 1, 2 and 4 are also possible for $A[4]$, but the smallest one is 1. Therefore, $A[4] = 1$.

Given a non-negative integer n, write a program to output $A[n]$.

Input

Your program is to read from standard input. The input consists of one line containing one non-negative integer n ($0 \leq n \leq 1000$).

Output

Your program is to write to standard output. Print exactly one line. The line should contain $A[n]$.

Examples

standard input	standard output
5	2
8	4
100	4

Problem B. Low Diameter Separator

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

You are given a tree with N vertices. Let's call a subset S of vertices a "low diameter separator" if, after deleting all vertices in S from the tree, the resulting forest has diameter strictly less than K. Compute the number of distinct low diameter separators.

The diameter of a forest is the maximum shortest distance over all pairs of vertices which are in the same connected component. If a vertex is removed, all edges incident with it are also removed. A forest of no vertices has diameter zero.

Input

The first line contains two integers N, K $(1 \le K \le N \le 300\,000)$

The next $N - 1$ lines contain two integers x_i, y_i denoting a pair of vertices connected by an edge. $(1 \le x_i, y_i \le N, x_i \ne y_i)$

The given graph is guaranteed to be a tree.

Output

Output the number of low diameter separators modulo $10^9 + 7$.

Examples

standard input	standard output
4 1 1 2 2 3 3 4	8
5 3 1 2 1 3 1 4 1 5	32
7 3 1 2 2 3 2 4 3 6 3 5 6 7	104

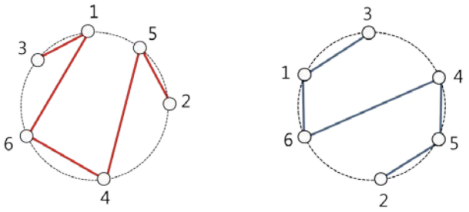
Problem C. Islands

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

There are twin islands with convex coastlines in the southern sea of the Korean Peninsula. There are n villages on the coastline in each island. All villages are numbered from 1 to n. These numbers have been randomly assigned to villages regardless of their locations. A sequence of the village numbers in clockwise direction on the coastline of an island is called a coastline sequence. Considering a village as a point, the closed polyline connecting all villages in order of a coastline sequence forms the boundary of a convex polygon.

At a meeting of village representatives from both islands, they decided to construct a road inside each island satisfying the following three conditions.

Each road passes through all villages of each island exactly once. Each road is not self-intersecting. Two road sequences of both islands are same, where a road sequence is a sequence of the village numbers along the road from the start to the end. For example, suppose that there are six villages in each island and that two coastline sequences of two islands are $(1, 5, 2, 4, 6, 3)$ and $(3, 4, 5, 2, 6, 1)$, respectively. Then two roads with a road sequence $(3, 1, 6, 4, 5, 2)$ satisfy the above conditions. See the figure below. If two coastline sequences are (1, 2, 3, 4, 5) and (1, 3, 5, 2, 4), there is no road sequence which satisfies the above conditions.



Given two coastline sequences of two islands, write a program to find a road sequence satisfying the above conditions if it exists.

Input

Your program is to read from standard input. The input starts with a line containing an integer n ($5 \le n \le 10^5$), where n is the number of villages in each island. The villages are numbered from 1 to n. In the following two lines, each line contains n integers which represent a coastline sequence of each island.

Output

Your program is to write to standard output. Print exactly one line. The line should contain n integers which represent a road sequence satisfying above conditions if it exists, otherwise -1. If there are one or more solutions, then print an arbitrary one.

Examples

standard input	standard output
6 1 5 2 4 6 3 3 4 5 2 6 1	3 1 6 4 2 5
5 1 2 3 4 5 1 3 5 2 4	-1

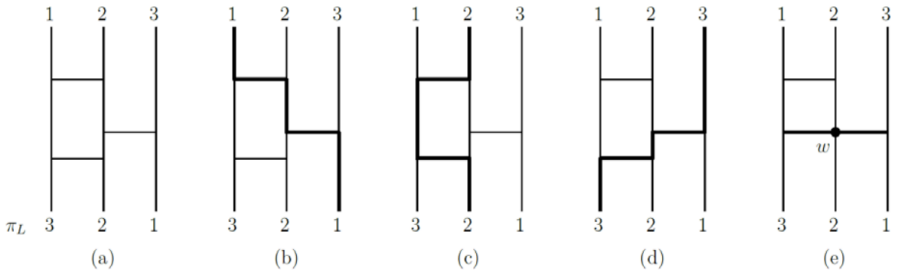
Problem D. Ladder Game

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

Consider a situation that we need to distribute n gifts to n people fairly and randomly. For this purpose one ancient technique is popular in Asia, and is usually used to represent a random permutation. Chinese call it Ghost Leg, Japanese Budda Lots, and Korean Ladder Game. We first start with some terms. This ladder consists of several vertical poles and horizontal bars connecting two adjacent vertical poles. From the top of each vertical pole, a path is traced through the ladder using the following three steps:

1. When tracing a vertical pole, continue downwards until an end of the first bar is reached, then continue along the bar.
2. When tracing a bar, continue along it until the end of the bar is reached, then continue down the vertical pole.
3. Repeat Step 1 and Step 2 until the bottom of a vertical pole is reached.

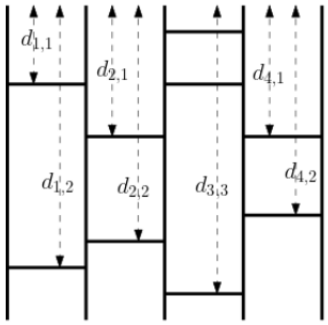
Figure (a) shows a ladder L with three vertical poles and three bars. The vertical poles are numbered with $1, \ldots, n$ from left to right. The paths of tracing these three vertical poles are shown in (b), (c), and (d). The input (1, 2, 3) is permuted finally to $p_L = (3, 2, 1)$ by the ladder L. Note that we do not allow the case that two immediately adjacent horizontal bars meet at some point (like w in (e)) because there is no unique way in tracing at the point.



You are given a ladder L which achieves a permutation p_L. If the permutation p_L is unchanged after removing a set of bars containing a particular bar, then the bar is said to be redundant for p_L. Your task is to find all redundant bars for p_L. This is equivalent to constructing a minimal ladder by removing all redundant bars from L. In the minimal ladder of L, the removal of any non-empty set of bars from the ladder gives a different permutation from p_L.

Input

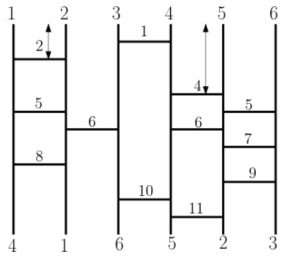
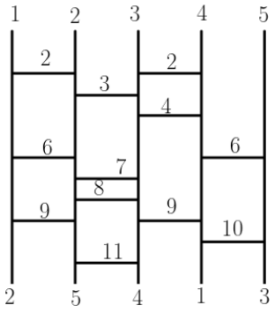
Your program is to read from standard input. The input starts with a line containing one integers n ($3 \le n \le 50$), where n is the number of vertical poles. The vertical poles are ordered from left to right. Let $d_{i,j}$ denote the depth of the j-th bar between the i-th and $(i+1)$-th poles from the top, which is an integer between 1 and 1,000. In the following $n-1$ lines, the i-th line contains the sequence of depths $d_{i,j}$, where $1 \le i < n$, $j \ge 1$ and $d_{i,j} < d_{i,j+1}$. Notice that this depth sequence is ended with zero (0), which is not a depth but just a marker to indicate the end of the sequence. See figure below for illustration.



Output

Your program is to write to standard output. Print a set of the bars to be remained in a minimal ladder for p_L. The first line should contain k, the number of the bars remained in the minimal ladder. Then, each line of the following k lines should contain two indices i and j of $d_{i,j}$ of the bar in the minimal ladder. Note that the minimal ladder is not unique.

Examples

standard input	standard output	Notes
6 2 5 8 0 6 0 1 10 0 4 6 11 0 5 7 9 0	8 3 1 4 1 5 1 2 1 4 2 1 3 3 2 4 3	
5 2 6 9 0 3 7 8 11 0 2 4 9 0 6 10 0	6 1 1 3 1 2 1 4 1 3 3 2 4	

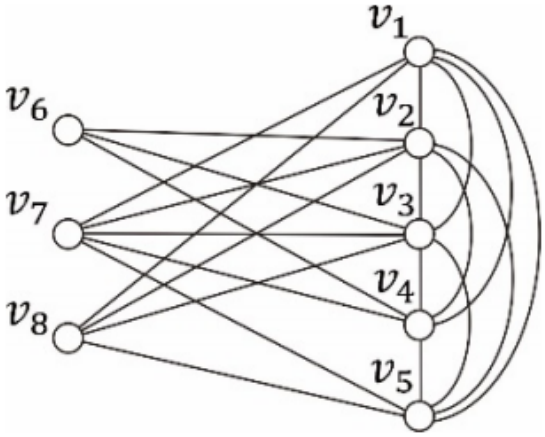
Problem E. Network Vulnerability

Input file:	*standard input*
Output file:	*standard output*
Time limit:	6 seconds
Memory limit:	512 mebibytes

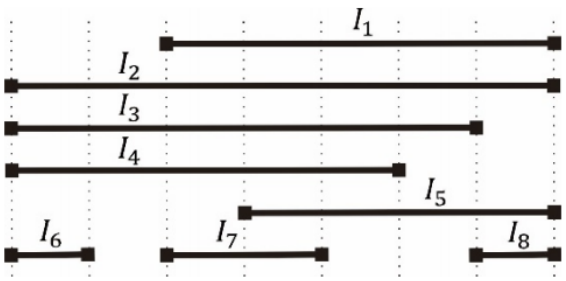
Network vulnerability refers to how much the network performance reduces in various cases of disruptions, such as natural disasters, element failures, or adversarial attacks. A network is frequently represented as a graph, in which the vertices and edges correspond to nodes and links, respectively. So the terms network and graph are used interchangeably here. The connectivity, which is defined to be the minimum number of vertices whose deletion results in a disconnected graph or a one-vertex graph, is recognized as the most important measure to evaluate the vulnerability of a network.

Nonetheless, the connectivity only partly reflects the resistance of a graph to the deletion of vertices. Accordingly, there have been many studies proposing other metrics to account for the network vulnerability, among which the toughness and the scattering number appear to be the most popular. The underlying notion of the toughness and scattering number is the maximum number of connected components resulting from removing k vertices from a graph. Let $c_k(G)$ denote the maximum number of connected components obtained by removing exactly k vertices from a graph G. It is unfortunate that given a graph G and a positive integer k, the problem of determining $c_k(G)$ is computationally intractable.

For some graph classes like interval graphs, however, $c_k(G)$ can be computed in polynomial time. An interval graph is the intersection graph of a family of (closed) intervals on the real line, where two vertices are connected with an edge if and only if their corresponding intervals intersect. The family is usually called an interval representation for the graph. See figures below for an example of an interval graph and its interval representation.



An example of the interval graph in usual form.



And this is an interval representation of the graph from the picture above.

Given an interval representation of an interval graph G with n vertices, your job is to write an efficient running program for computing $c_k(G)$ for all k included in $\{0, 1, \ldots, k-1\}$. If the interval representation shown in second figure, for example, is given, then $c_0(G)$, $c_1(G)$, $c_2(G)$, $c_3(G)$, $c_4(G)$, $c_5(G)$, $c_6(G)$, $c_7(G)$ are 1, 1, 1, 2, 2, 3, 2, 1, respectively.

Input

Your program is to read from standard input. The first line contains a positive integer n representing the number of intervals, where $n \leq 2,000$. In the following n lines, each contains a pair of left and right endpoints of an interval. You may assume that the endpoints, left or right, of intervals are integers between $-10^8 and 10^8$, inclusive.

Output

Your program is to write to standard output. Print exactly one line which contains the sequence $c_0(G)$, $c_1(G)$, …, $c_{n-1}(G)$ of n numbers separated by a single space.

Examples

standard input	standard output
8 3 8 1 8 1 7 1 6 4 8 1 2 3 5 7 8	1 1 1 2 2 3 2 1
3 -2 -2 -2 7 -2 7	1 1 1
4 -1 -1 3 4 5 6 7 8	4 3 2 1
5 1 2 2 3 3 4 6 7 7 8	2 3 3 2 1

Problem F. Quadrilaterals

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

A quadrilateral is a polygon with exactly four distinct corners and four distinct sides, without any crossing between its sides. A quadrilateral is called convex if all the inner angles at its corners are less than 180 degrees, or called non-convex, otherwise. See the illustration below for a convex quadrilateral (left) and a non-convex quadrilateral (right).

In a test problem, you are given a set P of n points in the plane, no three of which are collinear, and asked to find as many quadrilaterals as possible by connecting four points from P, while each point in P can be used limitlessly many times and those quadrilaterals you find may freely overlap each other. You will earn different credits for each quadrilateral you find, depending on its shape and area. In principle, you earn more credits for convex quadrilaterals and for quadrilaterals with minimum area.

More precisely, the rules for credits are as follows, where a denotes the minimum over the areas of all possible quadrilaterals formed by connecting four points in P:

- For each distinct convex quadrilateral with area exactly a, you earn 4 credits.
- For each distinct non-convex quadrilateral with area exactly a, you earn 3 credits.
- For each distinct convex quadrilateral with area strictly larger than a, you earn 2 credits.
- For each distinct non-convex quadrilateral with area strictly larger than a, you earn 1 credit.

Note that two quadrilaterals are distinct unless the corners and sides of one are exactly the same to the other's, and that there may be two or more quadrilaterals of the minimum area a.

You of course want to find all possible quadrilaterals and earn the maximum possible total credits. Given a set P of n points in the plane, write a program that outputs the maximum possible total credits you can earn when you find all possible quadrilaterals from the set P.

Input

Your program is to read from standard input. The input starts with a line containing an integer n ($4 \le n \le 1\,000$), where n denotes the number of points in the set P. In the following n lines, each line consists of two integers, ranging $-10^9 to 10^9$, representing the coordinates of a point in P. There are no three points in P that are collinear.

Output

Your program is to write to standard output. Print exactly one line consisting of a single integer that represents the maximum possible total credits you can earn from the set P.

Examples

standard input	standard output
4 0 0 1 0 0 1 1 1	4
4 0 0 10 0 5 10 5 8	5
4 0 0 10 0 5 10 5 3	7
5 0 0 0 5 5 0 5 5 4 2	14

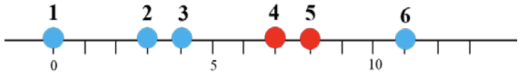
Problem G. Same Color

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

There are n distinct points with m colors on the line, where $m \le n$. Let S be the set of those points. We want to select a non-empty subset C of S that satisfies the following:

For every point p in $S - C$, not belonging to C, the closest point of p among points in C has the same color as p. Of course, if there are more than one closest point of p in C, then it is sufficient that one of them has the same color as p.

For example, there are six points labeled by 1 to 6 with two colors in figure below. Points 4 and 5 are red and the others blue. The set $\{2, 4, 6\}$ satisfies the above property. But the set $\{2, 4\}$ does not satisfy the property, because the closest point of point 6 in $2, 4$ is point 4, which is different from the color of point 6. In fact, the set $\{2, 4, 6\}$ is a minimum cardinality subset to satisfy the property.



Given n distinct points on the line and m their colors, write a program to find a non-empty subset C of S with minimum cardinality satisfying the above property and to output the minimum cardinality.

Input

Your program is to read from standard input. The input starts with a line containing two integers, m and n ($1 \le m \le n \le 10^5$), where m is the number of colors and n is the number of points. The points are numbered 1 to n from left to right on the line, and the colors are numbered 1 to m. The second line contains a sequence of sorted n integers in increasing manner, where the i-th number is the coordinate of the point i. The coordinates x of points satisfy $0 \le x \le 10^9$ and are all distinct. The third line contains a sequence of n integers, where the i-th number is the color of the point i, which is between 1 and m.

Output

Your program is to write to standard output. Print exactly one line. The line should contain the minimum cardinality of a non-empty subset C of S to satisfy the above property.

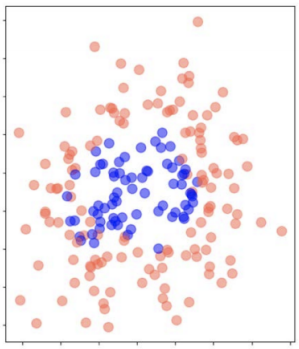
Examples

standard input	standard output
2 6 0 3 4 7 8 11 1 1 1 2 2 1	3
2 6 0 3 4 7 8 11 1 2 1 2 2 1	5

Problem H. Strike Zone

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

The strike zone in baseball is the volume of space which a baseball must pass through in order to be called a strike, if the batter does not swing. A baseball that misses the strike zone is called a ball, if the batter does not swing. Figure below shows the locations of baseballs at plate which were captured by a ball tracking device during a baseball match. Each blue point was called a strike and each red point was called a ball during the match. This may motivate us to define a rectangular region that represents the strike zone of the match, by analyzing such a ball tracking data of the match.



In this problem, you are given two sets, P_1 and P_2, of points in the plane and two positive constants c_1 and c_2. You are asked to find an axis-parallel rectangle R that maximizes the evaluation function $eval(R) = c_1 \cdot s - c_2 \cdot b$, where s is the number of points in the intersection of P_1 and R and b is the number of points in the intersection of P_2 and R.

Input

Your program is to read from standard input. The input starts with a line containing an integer n_1 ($1 \le n_1 \le 1000$), where n_1 denotes the number of points in P_1. In the following n_1 lines, each line consists of two integers, ranging -10^9 to 10^9, representing the coordinates of a point in P_1. The next line contains an integer n_2 ($1 \le n_2 \le 1000$), where n_2 denotes the number of points in P_2. In the following n_2 lines, each line consists of two integers, ranging -10^9 to 10^9, representing the coordinates of a point in P_2. There are no two points in the union of P_1 and P_2 that share the same x or y coordinate. Then the next line consists of two integers, c_1 and c_2, ranging 1 to 10^4.

Output

Your program is to write to standard output. Print exactly one line consisting of one integer that is $eval(R)$, where R is an axis-parallel rectangle with the maximum possible eval value for P_1 and P_2 with respect to c_1 and c_2.

Examples

standard input	standard output
2 -1 -1 4 4 2 0 0 2 2 5 2	6
3 0 5 3 3 8 -1 3 1 4 6 0 7 1 3 2	4

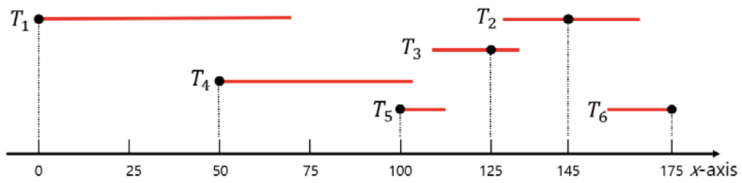
Problem I. Thread Knots

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

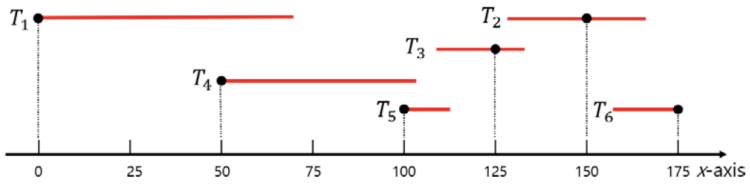
There are n threads on the x-axis. The length of the i-th thread, say T_i, is denoted by l_i and the location of its starting point by x_i, both being integers. We want to make a knot in each thread. The location of the knot must also be an integer. The knot can be made at any point in the thread and it is assumed that the length of the thread is not reduced by the knot. You can assume that no thread is totally contained by another, which means that there are no two thread T_i and T_j $(i \neq j)$ such that $x_j \leq x_i$ and $x_i + l_i \leq x_j + l_j$.

We want to determine the location of the knot for each thread in order to make the distance between the closest two neighboring knots as big as possible.

For example, the figures below show the locations of the knots for six threads. The location of a knot is denoted by a point. All the threads actually lie on the x-axis, however, they are drawn separately to distinguish each other. In figure below, the distance between the closest two knots is 20.



However, if we make the knot for T_2 at different location as shown in next figure, the distance between the closest two knots becomes 25, which is what this problem requests.



Given information about the n threads, write a program that calculates the maximum distance between two closest knots.

Input

Your program is to read from standard input. The input starts with a line containing one integer, n $(2 \leq n \leq 10^5)$, where n is the number of threads. In the following n lines, the i-th line contains two integers x_i $(0 \leq x_i \leq 10^9)$ and l_i $(1 \leq l_i \leq 10^9)$, where x_i and l_i denote the location of the starting point and the length of the i-th thread, respectively, You can assume that no thread is totally contained by another, which means that there are no two threads T_i and T_j $(i \neq j)$ such that $x_j \leq x_i$ and $x_i + l_i \leq x_j + l_j$.

Output

Your program is to write to standard output. Print exactly one line. The line should contain an integer which is the maximum distance between two closest knots.

Examples

standard input	standard output
6 0 67 127 36 110 23 50 51 100 12 158 17	25
6 0 40 10 55 45 28 90 40 83 30 120 30	30
3 0 20 40 10 100 20	50

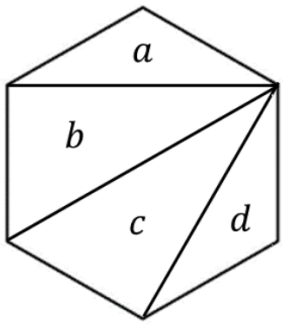
Problem J. Triangulation

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

A regular n-sided polygon P can be partitioned into $n-2$ triangles by adding non-crossing line segments connecting two vertices of P. For example, a square can be partitioned into two triangles, a regular pentagon can be partitioned into three triangles, and a regular hexagon can be partitioned into four triangles. The resulting set of triangles is called a triangulation of P. There exist two or more triangulations of P if n is greater than three.

Once a triangulation T of P is chosen, the distance between two triangles a and b in T is defined to be the number of hops crossing the borders of two adjacent triangles when you travel from a to b. Note that you must stay inside the polygon P, at any time during this travel, that is, it is not allowed to hop crossing the border of P.

For example, the distance of a and d in the triangulation shown in figure below is three since the triangles, a, b, c, and d, should be visited to travel from a to d, and you have to hop three times crossing borders between triangles.



The diameter of a triangulation T is the maximum of the distances between all pairs of triangles in T. Write a program to find a triangulation of a regular n-sided polygon P whose diameter is the minimum over all possible triangulations and print its diameter.

Input

Your program is to read from standard input. The input starts with a line containing n ($3 \le n \le 10^6$), where n is the number of sides of the regular n-sided polygon.

Output

Your program is to write to standard output. Print exactly one line. The line should contain the minimum diameter of the triangulations of a regular n-sided polygon.

Examples

standard input	standard output
3	0
4	1
6	2

Problem K. Washer

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

You have $n$clothes and a washer. The washer is large enough to wash all clothes at once. However, we should worry about the color transfer: if we put clothes of different colors in the washer, the dye from one may stain another. Precisely, let r_i, g_i, b_i denote the amount of red, green, blue color of the i-th clothes. When n clothes are washed together, the *color transfer* c is defined by

$$c = \sum_{i=1}^{n} (r_i - r)^2 + (g_i - g)^2 + (b_i - b)^2$$

where r, g, and b are the averages of r_i, g_i, b_i respectively. The i-th clothes with r_i, g_i, and b_i is defined as a point (r_i, g_i, b_i) in 3-dimensional RGB space. You can assume that no three points (clothes) are on a same line and no four points (clothes) are on a same plane in RGB space.

The washer consumes a lot of electricity, and you have to partition n clothes into at most k groups, and run the washer for each group. The total color transfer is the sum of color transfers from each run. Given the color information of n clothes and k, write a program to calculate the minimum total color transfer.

Input

Your program is to read from standard input. The first line contains two integers n ($1 \le n \le 100$) and k ($1 \le k \le 2$). In the following n lines, the i-th line contains three integers r_i, g_i, b_i ($0 \le r_i, g_i, b_i \le 1000$).

Output

Your program is to write to standard output. Print exactly one line containing the minimum total color transfer with absolute error 10^{-6} or better.

Examples

standard input	standard output
2 1 36 16 85 74 87 38	4347.000000
1 2 12 26 90	0.000000

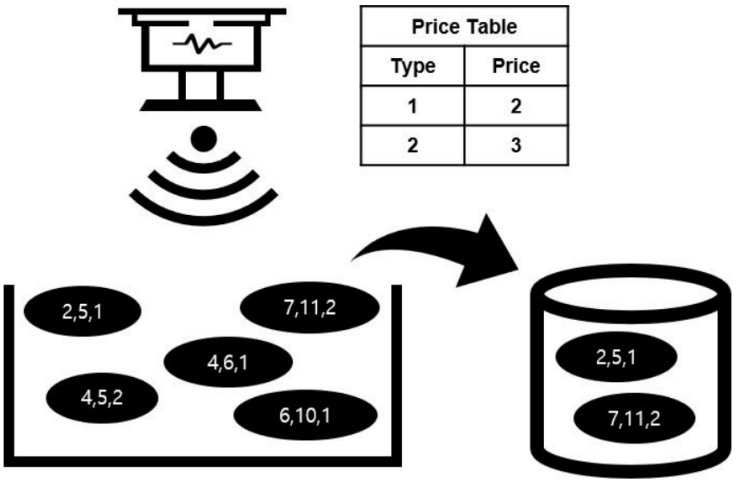
Problem L. What's Mine is Mine

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

The hot new video game "Mining Simulator" has just been released. In the game, rare earth mineral ores appear at certain times and you can mine them when they appear. After mining, you can later turn in the minerals for money. The quantity of mineral available during an appearance is proportional to the duration of the appearance and the price per unit of each mineral is decided beforehand.

The game contains a geological sensor that determines times when mineral ores will appear during a given day and at the beginning of each day you have a price list for each mineral. Assuming you mine out the mineral in exactly the amount of time that it is available, you cannot partially mine minerals (you must either not mine any of a given occurrence or mine all of it) and you can only mine one ore occurrence at a time.

Given a list of the prices of m minerals and n ore occurrences during a day, write a program to output the maximum amount of money you can earn from mining on that day. The mineral amount is the appearance time (end time - start time). You can mine an ore right after finishing the previous mining. In other words, one mined-ore's end time can be same as another mined-ore's start time. In the case depicted in the figure, if you choose the mineral of type 1 that appears at time 2 and disappears at time 5, then the mineral amount is $5 - 2 = 3$ and you earn $3 \times 2 = 6$. Next, if you choose the mineral of type 2 that appears at time 7 and disappears at time 11, then the mineral amount is $11 - 7 = 4$ and you earn $4 \times 3 = 12$. Therefore, you earn 18 in total.



Input

Your program is to read from standard input. The input starts with a line containing two integers, m and n ($1 \le m \le 100$, $1 \le n \le 10^4$), where m is the number of types of minerals and n is the number of ore occurrences during the day. The mineral types are labeled from 1 to m. The following m lines contain a single integer corresponding to the price of one unit of the i-th mineral type on that day (the price is between 1 and 10^4). The following n lines represent ore occurrences. Each line contains three integers, s, e, t where s is the start time, e is the end time and t is the mineral type in each ore occurrence with $0 < s < e < 15\,000$ and $1 \le t \le m$. The amount of mineral at each occurrence is $e - s$.

Output

Your program is to write to standard output. Print exactly one line. The line should contain the maximum amount of money that can be earned on that day.

Examples

standard input	standard output
2 5 2 3 2 5 1 4 5 2 4 6 1 7 11 2 6 10 1	18
3 5 2 3 1 1 4 1 3 6 3 5 8 2 7 10 1 9 12 2	24
5 7 1 2 3 4 5 1 5 2 3 8 1 2 4 3 3 9 2 4 10 5 7 11 4 5 7 3	36

Problem M. Alternative Facts

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

In the year 3020, the Divided Republic of Byteland had an inauguration ceremony of its 237th president. Well, it's not just one president, but many presidents: Byteland is divided into N separate entities, and each entity elects their own president.

Reporter Seohyun knows that A_i people had participated in the i-th president's ceremony. Now, he has to write an article about the ceremonies. In the old times when the world was oppressed by science and facts, writing an article was a simple thing. However, it is now the year 3020, and therefore we should pursue higher goals..

Given a sequence A of length N, a sequence B is an **alternative fact** if all of the following statements are true.

- The length of B is N.
- There exists a permutation P of size N such that $B_{P_i} = A_i$. In other words, if you sort the sequence A and B, then $A_i = B_i$ for all $1 \le i \le N$.
- For all $1 \le i \le L$, $|A_i - B_i| \le K$. K, L are given as an input. (Who reads the whole article?)

An alternative fact B is **greater** than C if there exists an index $1 \le i \le N$ such that $B_j = C_j$ for all $j < i$ and $B_i > C_i$. An alternative fact is **huge** if it is greater than all other alternative facts.

Your job is simple: Find the huge alternative fact of A.

Input

The first line contains three integers N, K, L ($1 \le N \le 200\,000, 0 \le K \le 10^9, 0 \le L \le N$)

The next line contains N integers $A_1, A_2, \ldots, A_N$ ($1 \le A_i \le 10^9$)

Output

Print the huge alternative fact of the sequence A on one line, separated by spaces.

Examples

standard input	standard output
5 3 5 7 6 1 9 4	9 7 4 6 1
5 2 5 7 6 1 9 4	9 6 1 7 4
5 2 3 7 6 1 9 4	9 7 1 6 4
5 2 0 7 6 1 9 4	9 7 6 4 1