Problem Set

Please check that you have 10 problems that are spanned across 23 pages in total (including Korean translation and this cover page).

A. Best Partition	(1+2 pages)	Korean translation available
B. Bridge	(2 pages)	
C. Docking Day	(2 pages)	
D. Fox Buki	(2 pages)	
E. Image Analysis	(2 pages)	
F. Inverse Look-and-Say	(1+2 pages)	Korean translation available
G. Magical Sort	(2 pages)	
H. Mountain	(2 pages)	
I. Quadratic Equation	(1+1 pages)	Korean translation available
J. Rescue Squad	(2 pages)	

Problem A
Best Partition
Time Limit: 1 Second

You are given two sequences, A and B which are permutations of integers between 1 and n (inclusive). You want to divide each of them into k partitions by picking $k-1$ positions, both from A and B. There is one rule: for each $i = 1, 2, \cdots, k$, the position of the minimum element within the i-th partition of A must be equal to the position of the minimum element within the i-th partition of B. For example, assume that $A = (1, 3, 2, 5, 4)$ and $B = (5, 4, 3, 2, 1)$. Then you can divide A into $(1), (3, 2), (5, 4)$, which means that you divide B into $(5), (4, 3), (2, 1)$. It meets the condition mentioned above. Of course, you can divide A and B into n partitions whose length is one. We want to find the partition with the smallest k satisfying the condition mentioned above and will call it ***the best partition***. You can see that the example above is a best partition.

Given n, A, and B, write a program to find the best partition and output k.

Input
Your program is to read from standard input. The input starts with a line containing an integer n ($1 \leq n \leq 3{,}000$) where n is the length of A and B. The second line contains n integers between 1 and n (inclusive) which describes A. The third line also contains n integers between 1 and n (inclusive) which describes B. Note that in the second and third lines, no integer appears more than once in the same line.

Output
Your program is to write to standard output. Print exactly one line. The line should contain the number k of partitions in the best partition.

The following shows sample input and output for three test cases.

Sample Input 1

```
5
1 3 2 5 4
5 4 3 2 1
```

Output for the Sample Input 1

```
3
```

Sample Input 2

```
5
1 2 3 4 5
1 5 4 3 2
```

Output for the Sample Input 2

```
1
```

Sample Input 3

```
5
1 2 3 4 5
5 4 3 2 1
```

Output for the Sample Input 3

```
5
```

Problem A

최적의 분할

제한 시간: 1 초

1 이상 n 이하 정수들로 이루어진 길이 n인 순열 A와 B가 주어진다. A와 B에서 동일한 $k-1$개의 위치를 골라서 각각 k개의 조각으로 나누려고 한다. 단, 각 $i = 1, 2, \cdots, k$에 대해, A의 i번째 조각의 최솟값의 위치와 B의 i번째 조각의 최솟값의 위치가 서로 같아야 한다. 예를 들어서, $A = (1, 3, 2, 5, 4)$ 이고 $B = (5, 4, 3, 2, 1)$ 라 하자. 만약, A를 $(1), (3, 2), (5, 4)$로 나누면, B는 $(5), (4, 3), (2, 1)$로 나누어지며, 위에서 설명한 조건을 만족시킨다. 물론 A와 B를 길이가 1인 n개의 조각들로 나누면 위 조건을 쉽게 만족시킬 수 있다. 따라서 우리는 조건을 만족하면서 조각의 수 k가 최소가 되도록 나누고자 하며, 이러한 분할을 ***최적의 분할***이라고 하자. 위 예시에서 $k = 3$인 분할이 최적의 분할이다.

n, A, B가 주어질 때, 최적의 분할을 찾고, 그 때의 k를 출력하는 프로그램을 작성하시오.

Input

입력은 표준입력을 사용한다. 첫 줄에 A와 B의 길이를 나타내는 양의 정수 n $(1 \le n \le 3{,}000)$이 주어진다. 두 번째 줄에 A에 대한 정보가 주어지며, 1 이상 n 이하인 n개의 정수들이 주어진다. 세 번째 줄에 B에 대한 정보가 주어지며, 1 이상 n 이하인 n개의 정수들이 주어진다. 두 번째 줄과 세 번째 줄에서, 같은 줄에는 같은 정수가 두 번 이상 주어지지 않는다.

Output

출력은 표준출력을 사용한다. 첫 줄에 최적의 분할의 조각 수를 출력한다.

다음은 세 테스트 경우에 대한 입출력 예이다.

Sample Input 1

```
5
1 3 2 5 4
5 4 3 2 1
```

Output for the Sample Input 1

```
3
```

Sample Input 2

```
5
1 2 3 4 5
1 5 4 3 2
```

Output for the Sample Input 2

```
1
```

Sample Input 3

```
5
1 2 3 4 5
5 4 3 2 1
```

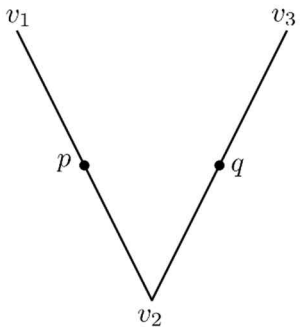
Output for the Sample Input 3

```
5
```

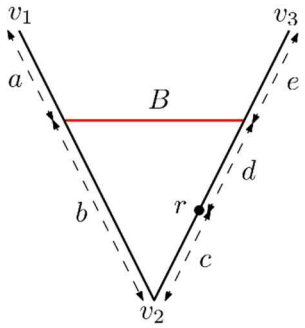
Problem B
Bridge
Time Limit: 1 Second

You are given a simple polygonal path $P = \langle v_1, v_2, \cdots, v_n \rangle$ with $x(v_i) \leq x(v_{i+1})$ for every $i = 1, \cdots, n-1$ in the plane. Every segment of P has a positive length and no two segments of P intersect, except at their endpoints. The distance between any two points p, q in P is the length of the path from p to q along P, that is, the sum of segment lengths of the path. The diameter of P is the maximum of all possible distances between two points in P.

For example, consider a polygonal path $P = \langle v_1, v_2, v_3 \rangle$ as shown in Figure (a) below. The distance between the two midpoints p, q of the segments is the sum of lengths of segments pv_2 and v_2q. The diameter of P is the distance between the two end vertices v_1, v_3 of P, that is the sum of lengths of segments v_1v_2 and v_2v_3.



(a)



(b)

Now we add a bridge to P. A bridge B of P is a segment parallel to the x-axis and connecting two points of P such that for every point z of B, except the endpoints of B, P has no point z' with $x(z) = x(z')$ and $y(z) \leq y(z')$, where $x(t)$ is the x-coordinate and $y(t)$ is the y-coordinate of a point t in the plane. Then a path connecting two points of P can use B by entering and exiting at the endpoints of B. Thus, the distance between two points of P is the length of the shorter path between the path using B and the path not using B.

For example, if we add a bridge B as shown in Figure (b) above, the distance between v_1 and v_3 is $a + e + |B|$ by the path using B, where $|B|$ is the length of B. The distance between v_1 and r is the smaller between the length $a + d + |B|$ of the path using B and the length $a + b + c$ of the path not using B.

Given a simple polygonal path $P = \langle v_1, v_2, \cdots, v_n \rangle$ with $x(v_i) \leq x(v_{i+1})$ for every $i = 1, \cdots, n-1$, write a program to output the infimum (greatest lower bound) of diameters of P using a bridge.

Input

Your program is to read from standard input. The first line contains the number n ($3 \leq n \leq 10^5$) of vertices of $P = \langle v_1, v_2, \cdots, v_n \rangle$ with $x(v_i) \leq x(v_{i+1})$ for every $i = 1, \cdots, n-1$. In the next n lines, the i-th line contains the x-coordinate $x(v_i)$ and the y-coordinate $y(v_i)$ of v_i $(-100{,}000 \leq x(v_i), y(v_i) \leq 100{,}000)$. All the coordinates are integers, and no two vertices are at the same position.

Output

Your program is to write to standard output. Print the infimum z of diameters of P using a bridge. If no bridge can be placed on P, print the diameter of P with no bridge. The output z should be in the format that consists of its integer part, a decimal point, and its fractional part, and will be decided to "correct" if it holds that $\frac{|a-z|}{a} < 10^{-6}$, where a denotes the exact answer.

The following shows sample input and output for two cases.

Sample Input 1	Output for the Sample Input 1
3 0 3 3 0 6 3	6.56301792813632

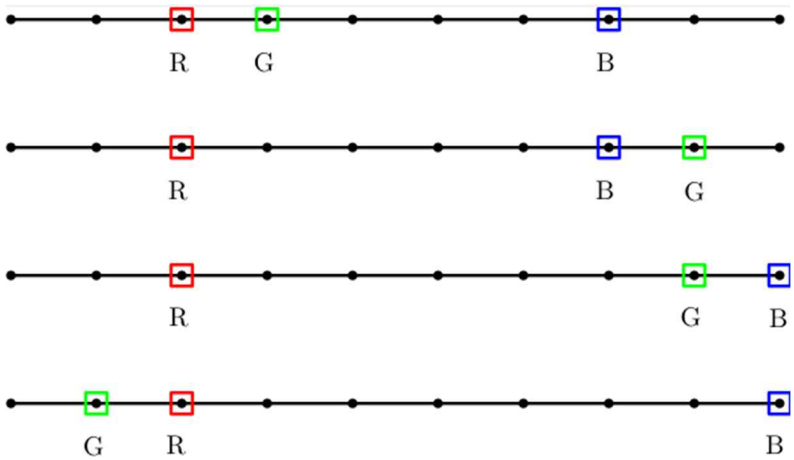
Sample Input 2	Output for the Sample Input 2
4 0 1 0 0 1 0 1 1	2.0

Problem C
Docking Day
Time Limit: 2 Seconds

A space station has docking ports labeled by distinct positive integers $1, 2, 3, \cdots$ arranged in a straight line. Port 1 is the leftmost, and the line extends infinitely to the right. Three labeled ships—Red (R), Green (G), and Blue (B)—are currently at different ports. Due to maintenance, traffic control must re-dock the three ships to newly assigned target ports. To keep clear sight lines and safe spacing during re-docking, the moving ship must pass over exactly one other ship—no more, no less. Specifically, traffic control wants to re-dock while satisfying these constraints:

1. Each ship must end at its own target port.
2. At any time, no two ships may occupy the same port.
3. In one move, choose one ship and place it on an empty port so that exactly one of the other two ships has a port strictly between the old and new ports.

For example, suppose R, G, and B are currently at ports $3, 4, 8$ and their target ports are $3, 2, 10$, respectively. In three moves - (1) move G from 4 to 9 (passing B), (2) move B from 8 to 10 (passing G), and (3) move G from 9 to 2 (passing R) - all three ships reach their targets. See the figures below.



Given the current ports and target ports of the three ships, write a program to compute the minimum number of moves required to re-dock them to the target ports.

Input

Your program is to read from standard input. The input starts with a line containing three distinct integers, r_1, g_1 and b_1 $(1 \leq r_1, g_1, b_1 \leq 10^6)$, which denote the positions of the current ports of R, G, and B, respectively. The following line contains three distinct integers, r_2, g_2 and b_2 $(1 \leq r_2, g_2, b_2 \leq 10^6)$, which denote the positions of the target ports of R, G, and B, respectively.

Output

Your program is to write to standard output. Print exactly one line. The line should contain the minimum number of moves required to re-dock them to the target ports.

The following shows sample input and output for two test cases.

Sample Input 1	Output for the Sample Input 1
3 4 8 3 2 10	3

Sample Input 2	Output for the Sample Input 2
3 4 5 6 2 1	3

Problem D
Fox Buki
Time Limit: 2 Seconds

In celebration of the galaxy-famous idol *Fox Buki*, a group of enthusiastic fans is curating her trading card collection. There are n fans and n^2 distinct cards labeled 0 through n^2-1. The type of card i is $\lfloor i/n \rfloor$, the integer quotient when i is divided by n, so each type $t \in \{0, 1, \cdots, n-1\}$ appears in exactly n cards.

Initially, the n^2 cards are partitioned among the fans so that each fan holds exactly n cards. Starting from the initial distribution, the fans aim to rearrange the cards by performing a sequence of valid swaps so that every fan ends up with exactly one card of each type.

A swap is a pair (a, b) of integers with $0 \le a < b \le n^2 - 1$; executing it swaps the cards labeled a and b. When the swap occurs, the two cards must be held by different fans.

To reduce wear and tear on the corners of the cards during the swap processing, the fans need to follow the preservation rule: no single card should be involved in too many swaps. For each card c, let $\text{usage}(c)$ be the number of swaps involving c. Among all sequences of valid swaps that end with each fan holding exactly one card of each type, minimize $\max_{0 \le c \le n^2-1} \text{usage}(c)$. Note that the total number of swaps need not be minimized.

Given the labels of the cards initially held by n fans, write a program to output such a sequence of swaps. It is shown that there is such an optimal sequence of at most n^3 swaps.

Input
Your program is to read from standard input. The first line contains an integer n $(1 \le n \le 100)$.

The following n lines describe the initial holdings. The i-th line among these n lines contains n distinct integers—the labels of the cards initially held by i-th fan—listed in increasing order. These n lists of integers form a partition of $\{0, 1, \cdots, n^2 - 1\}$.

Output
Your program is to write to standard output. Print an integer k $(0 \le k \le n^3)$, the number of swaps.

Then print k lines, each containing two integers a and b $(0 \le a < b \le n^2 - 1)$, describing the swaps (a, b) in order. After performing these swaps:

- Each fan should hold exactly one card of each type.
- The maximum per-card usage should be the minimum.

If there are multiple valid solutions, anyone will be accepted.

The following shows sample input and output for three test cases.

Sample Input 1	Output for the Sample Input 1
2 0 1 2 3	1 1 2

Sample Input 2	Output for the Sample Input 2
2 0 3 1 2	0

Sample Input 3	Output for the Sample Input 3
3 0 2 8 1 6 7 3 4 5	3 2 3 4 6 0 1

Problem E
Image Analysis
Time Limit: 2 Seconds

A research laboratory is working on a project involving the automatic analysis of digital images.

The image under study is extremely large and is modeled as an $n \times n$ integer grid. The lower-left corner of the grid is $(1, 1)$ and the upper-right corner is (n, n). Among all the (positive) integer grid points, only p points are marked as *active*, each associated with a color ID. These active points are of particular interest to the researchers because they represent features extracted from the image—such as highlighted regions, detected objects, or important signals. It is guaranteed that all x-coordinates of the active points are distinct, and all y-coordinates of the active points are distinct.

To better understand the distribution of these features, the researcher analyzes the image using a rectangular *query frame* of fixed size $W \times H$, where W is the width and H is the height. The query frame must be entirely contained within the $n \times n$ grid and must align exactly with the grid. The researcher observes all the active points within the query frame and records their color IDs.

The task is not simply to count colors in the query frame, but to measure how **balanced** they are. In particular, the researcher is interested in the number of distinct color IDs that appear with frequencies in some *frequency range* $[A, B]$, i.e., lying inclusively between two thresholds A and B. A color that appears too rarely or too frequently in the frame may not be considered significant, while a color that occurs within a moderate frequency range is more important. Thus, we want to count the colors of the active points in the query frame whose frequencies are in the given frequency range.

Figure 1 illustrates an example on a 11×11 integer grid. In Figure 1 (a), ten active points are marked with four colors: gold (1), blue (2), red (3), and gray (4). In Figure 1 (b), a query frame of size $W = 5$ and $H = 6$ is placed with the lower-left corner at $(3, 3)$.

This query frame contains the active points of the colors gold, blue, red, and gray. For a frequency range $[A, B] = [2, 4]$, we calculate the frequency of each color only inside the query frame and select the ones whose frequencies lie in $[2, 4]$. In this example, the colors with frequencies in $[2, 4]$ are gold and blue because gold and blue appear twice, but the red and gray appear only once. Thus, the answer for this query frame is 2.

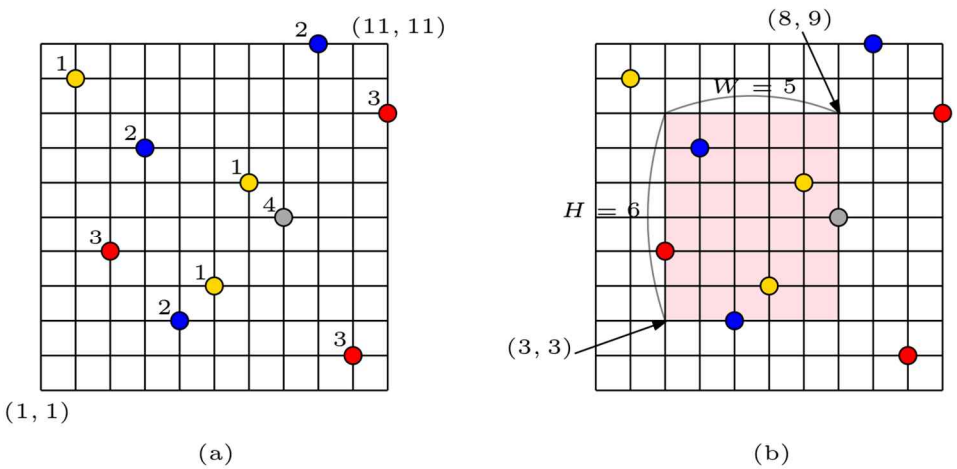


Figure 1. (a) Active points in $\mathbf{11 \times 11}$ grid with four colors.
(b) A query frame with $\boldsymbol{W = 5}$, $\boldsymbol{H = 6}$ with lower-left corner at $\mathbf{(3, 3)}$.

Given active points and their colors, you are asked to process q such queries. Each query specifies a query frame and a frequency range. For each query, you must output how many distinct color IDs inside the query frame have their frequencies lying in the frequency range.

Input

Your program is to read from standard input. The input starts with a line containing three integers n, p, and q ($2 \le n \le 2 \times 10^5$; $1 \le p \le 2 \times 10^5$; $1 \le q \le 2 \times 10^5$), where n is the size of the image, p is the number of active points, and q is the number of queries.

The second line contains two integers W and H ($1 \le W, H \le n - 1$), the width and height of the query frame, respectively.

Each of the next p lines contains three integers x, y, c ($1 \le x, y \le n$; $1 \le c \le 10^9$), describing an active point with color ID c located at the integer grid point (x, y). All x-coordinates are distinct, and all y-coordinates are distinct.

Each of the following q lines contains four integers L, D, A, B ($1 \le L \le n - W$; $1 \le D \le n - H$; $1 \le A \le B \le p$). This represents a rectangle $[L, L + W] \times [D, D + H]$, which is a query frame, and the frequency range $[A, B]$.

Output

Your program is to write to standard output. Print q lines. The i-th line should contain the number of distinct color IDs of the active points in the i-th query frame whose frequencies lie in $[A, B]$.

The following shows sample input and output for two test cases.

Sample Input 1

```
5 5 3
2 3
1 1 1
2 4 2
3 2 2
4 5 3
5 3 4
1 1 1 2
2 2 1 1
3 1 2 3
```

Output for the Sample Input 1

```
2
1
0
```

Sample Input 2

```
6 3 3
4 4
1 1 1
4 2 5
2 6 5
1 1 1 3
1 2 1 1
2 2 2 2
```

Output for the Sample Input 2

```
2
0
1
```

Problem F
Inverse Look-and-Say
Time Limit: 1 Second

Consider the following sequence: $2 \rightarrow 12 \rightarrow 1112 \rightarrow 3112 \rightarrow 132112 \rightarrow 1113122112 \rightarrow \cdots$.

The initial value of this sequence is 2, and subsequent values are generated as follows:
- 2 contains "one 2," so the next term is 12.
- 12 contains "one 1 and one 2," so the next term is 1112.
- 1112 contains "three 1's and one 2," and the next term is 3112.
- 3112 contains "one 3, two 1's, and one 2," the next term is 132112.

This sequence follows the "look-and-say" rule. Let $f(x)$ denote the number obtained by applying the rule of the sequence to a positive integer $x > 0$. Then, $f(2) = 12$, $f(12) = 1112$, $f(1112) = 3112$, $f(3112) = 132112$, and so on.

The function $f(x)$ is defined only when no digit in the positive integer x appears consecutively more than 9 times. By applying the "look-and-say" rule, we can obtain a number n such that $f(x) = n$. Your task is, given a positive integer n as input, to find an x such that $f(x) = n$. If x exists, it is unique. Note that for some positive integers n, no such x exists. For example, there is no positive integer x such that $f(x) = 311$. Likewise, there is no positive integer x such that $f(x) = 1111$. In the case of 1111, one might interpret it as "one 1, one 1" and think $x = 11$, but $f(11) = 21$. Thus, no positive integer can produce 1111 according to the "look-and-say" rule.

Input

Your program is to read from standard input. The input has a line containing one positive integer n less than $10^{1,000}$.

Output

Your program is to write to standard output. Print exactly one line with one integer x such that $f(x) = n$. If such x does not exist, print -1.

The following shows sample input and output for three test cases.

Sample Input 1	Output for the Sample Input 1
132112	3112

Sample Input 2	Output for the Sample Input 2
331	-1

Sample Input 3	Output for the Sample Input 3
1111	-1

Problem F
Inverse Look-and-Say
제한 시간: 1 초

다음과 같은 수열을 생각해보자: $2 \to 12 \to 1112 \to 3112 \to 132112 \to 1113122112 \to \cdots$.

이 수열의 초기값은 2이고, 이후의 수들은 다음과 같이 생성된다.

- 2에는 "1개의 2"가 있으므로, 다음 값은 12이다.
- 12에는 "1개의 1, 1개의 2 "가 있으므로 다음 값은 1112이다.
- 1112에는 "3개의 1, 1개의 2"가 있으므로 다음 값은 3112이다.
- 3112에는 "1개의 3, 2개의 1, 1개의 2"가 있으므로, 다음 값은 132112이다.

이 수열은 이러한 "look-and-say" 규칙으로 생성된다. 이 수열에서 다음 항을 계산하는 규칙을 양의 정수 $x > 0$에 적용해서 나오는 수를 $f(x)$라고 하자. 즉, $f(2) = 12$, $f(12) = 1112$, $f(1112) = 3112$, $f(3112) = 132112$와 같이 주어지는 양의 정수에 대해 이를 십진법으로 읽었을 때 연속되는 숫자의 개수를 보이는 대로 읽어서 만들어지는 수를 의미한다.

양의 정수 x에 연속해서 나타나는 숫자가 9회를 넘지 않을 때에만 $f(x)$가 정의되며, "look-and-say" 규칙을 적용하여 $f(x) = n$인 n을 얻을 수 있다. 여러분이 할 일은 입력으로 양의 정수 n이 주어질 때, $f(x) = n$인 양의 정수 x를 구하는 것이다. x가 존재한다면 그 값은 유일하다. 주의할 점은 어떤 양의 정수 n은 그러한 x를 가지지 않는다는 것이다. 예를 들어, $f(x) = 311$인 양의 정수 x는 존재하지 않는다. 또한 $f(x) = 1111$인 양의 정수 x 역시 존재하지 않는다. 1111의 경우 "1개의 1, 1개의 1"로 해석하여 x가 11이라 생각할 수 있으나, $f(11) = 21$이다. 어떤 양의 정수도 이 "look-and-say" 규칙에 따라 1111을 생성할 수 없다.

Input

입력으로 첫 줄에 $10^{1,000}$보다 작은 양의 정수 n이 주어진다.

Output

$f(x) = n$ 인 양의 정수 x가 존재한다면 그 값을 출력하고, 그렇지 않다면 -1을 출력한다.

다음은 세 테스트 경우에 대한 입출력 예이다.

Sample Input 1	Output for the Sample Input 1
132112	3112

Sample Input 2	Output for the Sample Input 2
331	-1

Sample Input 3	Output for the Sample Input 3
1111	-1

Problem G
Magical Sort
Time Limit: 3 Seconds

Once upon a time, there was a magician who had studied computer science for several years. One day, he was deeply impressed by the radix sort since it felt like real magic to him. He decided to use a similar mechanism to invent a new card shuffling magic trick. However, he had to make it more sophisticated so that no one would notice anything until the very end of the performance. Here's the trick he wrote down in his notebook:

1) There are a pile of cards and n assistants.
2) Let someone from the audience do the following:
   2a) Choose any integer $K \geq 1$.
   2b) Take any number of cards from the pile and write down one K-digit binary number on the front side of each of the cards. The same binary number may be written on two or more cards.
   2c) Distribute the cards to the n assistants in any arbitrary way. Note that the assistants might receive a different number of cards, possibly even zero.
3) For $k = 1, \cdots, K$, repeat the following:
   3a) The assistants pick up the pile of cards in front of themselves.
   3b) I ask each assistant in the pre-specified "work order" to do the following:
      (i) Divide the cards in his hand into two groups according to the k-th least significant bit, i.e., Group-0 for the cards with 0-bit and Group-1 for those with 1-bit, while preserving their original order.
      (ii) Stack the cards in front of other assistants (or possibly themselves) according to the "distribution table" which describes who gives which group of cards to whom (see, e.g., Table 1). The cards should be faced down so that the cards in front of each assistant can be collected in the work order of the assistants who gave the cards to them.

Table 1. Example of a distribution table

Assistant	ADA	BOB	JOHN	MAX	ZOE
Work order	5	2	3	1	4
Receiver of Group-0 cards	JOHN	MAX	JOHN	BOB	JOHN
Receiver of Group-1 cards	ADA	ZOE	ZOE	ZOE	ZOE

4) After K rounds, I collect the cards in the work order of assistants.

Since the audience may choose any K and write any length-K binary numbers without control, the magician carefully designed the distribution table and the work order so that the binary numbers can always be sorted in non-decreasing lexicographical order at the end of the above procedure regardless of the initial configuration.

The magician had a rehearsal with the distribution table in Table 1. At the beginning $K = 3$ was chosen each assistant received the cards as described in the first two rows in Table 2; note that the assistants are written in their work order, in which they perform Step-3b described above. The third and fourth rows of Table 2 show how the cards were divided at the first round. For example, MAX divided the cards into Group-0 (010, 100) and Group-1 (111, 011), after which he put Group-0 cards in front of BOB, and Group-1 cards in front of ZOE.

Table 2. Round 1 of the rehearsal

Assistant	MAX	BOB	JOHN	ZOE	ADA
Cards (initial)	010, 111, 011, 100	001, 000	-	000	110, 111
Group-0 cards	010, 100	000	-	000	110
Group-1 cards	111, 011	001	-	-	111
Cards after Round 1	000	010, 100	000, 110	111, 011, 001	111

The last row of Table 2 shows the stacked cards in front of each assistant at the end of the first round. For example, ZOE received three cards (111, 011, 001): two cards (111, 011) from MAX, and the other card (001) from BOB. And they were collected in the work order of the assistants who gave the cards.

The other two rounds were performed similarly, and the following table shows the cards stacked in front of each assistant at the end of the two rounds. Notice that after Round 3, one can obtain the binary numbers in non-decreasing lexicographical order by collecting the cards in work order of the assistants.

Assistant	MAX	BOB	JOHN	ZOE	ADA
Cards after Round 2	100	000	000, 001	010, 110, 111, 011	111
Cards after Round 3	000	-	000, 001, 010, 011	100, 110, 111	111

Once day, the magician noticed that there might exist more than one work orders for the same distribution table with which the above procedure always ends up with the binary numbers sorted correctly. For example, the procedure still works well even if BOB and MAX are swapped in the work order. The magician wants to know how many such work orders exist for his distribution table.

Given a distribution table, write a program to find the number of work orders with which the procedure works correctly in every case, regardless of the value of K and the binary numbers written on the cards. It is guaranteed that at least one such work order exists for the distribution table.

Input

Your program is to read from standard input. The input starts with a line containing an integer n ($2 \leq n \leq 10^5$), where n is the number of assistants. Then the distribution table is given in the following n lines. Each line contains the names of three assistants X, Y, Z, meaning that at each round k, X gives the cards with 0-bit at the k-th least significant bit to Y, and those with 1-bit to Z. Each name is a non-empty string consisting of up to four English upper letters. Every assistant appears as a receiver at least once in the table.

Output

Your program is to write to standard output. Print exactly one number W modulo $101{,}287$ where $W \geq 1$ is the number of work orders with which it is always possible to sort equal-length binary numbers in non-decreasing lexicographical order using the described procedure with the given distribution table.

The following shows sample input and output for three test cases.

Sample Input 1

```
5
ADA JOHN ADA
BOB MAX ZOE
JOHN JOHN ZOE
MAX BOB ZOE
ZOE JOHN ZOE
```

Output for the Sample Input 1

```
2
```

Sample Input 2

```
2
ADA BOB ADA
BOB BOB ADA
```

Output for the Sample Input 2

```
1
```

Sample Input 3

```
8
A A E
B A G
C A F
D A H
E A H
F C H
G B H
H D H
```

Output for the Sample Input 3

```
4
```

Problem H
Mountain
Time Limit: 2 Seconds

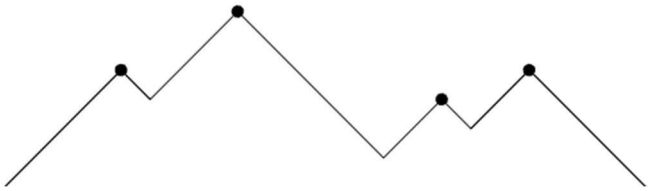


Figure 1 A mountain with four peaks

See Figure 1. What do you see? Yes, it is a mountain. Let's call such a figure a mountain. To be more precise, a mountain is an x-monotone polygonal chain consisting of at least two line segments whose slopes alternate between $+1$ and -1; the leftmost segment is a half-line (ray) of slope $+1$, and the rightmost segment is a half-line of slope -1. Every mountain has one or more peaks, which are endpoints incident to a segment of slope $+1$ to the left and a segment of slope -1 to the right; in other words, a peak is a locally highest point. In Figure 1, you can see a mountain consisting of eight segments, and it has four peaks, marked by small dots.

In this problem, you are given a set P of n points in the plane as input, and your task is to find a mountain with the maximum number of peaks that are chosen from P.

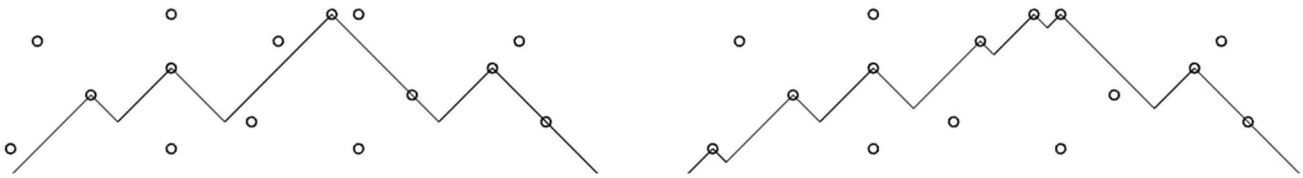


Figure 2 Two mountains with four and seven peaks, respectively, chosen from a given set $\boldsymbol{P}$ of 15 points

An example is illustrated in Figure 2, in which the input set P of $n = 15$ points, marked by small circles, is given, and to the left you can see a mountain with four peaks that belong to P, while to the right there is a mountain with seven peaks that belong to P. Since there is no such mountain with eight peaks chosen from P, the one to the right is a correct answer for the problem. See Sample Input/Output 2 below.

Input
Your program is to read from standard input. The input starts with a line containing a single integer n $(1 \leq n \leq 500{,}000)$, where n is the number of points in the input set P of points in the plane. In each of the following n lines, given are two integers x and y, both ranging from -10^6 to 10^6, inclusively, that represent the x- and y-coordinates of an input point (x, y) in P. You may assume that no two input points have the same coordinates.

Output
Your program is to write to standard output. Print exactly one line. The line should contain an integer that represents the number of peaks of a mountain with the maximum number of peaks that are chosen from P.

The following shows sample input and output for three test cases.

Sample Input 1	Output for the Sample Input 1
4 0 0 1 0 2 0 3 0	4

Sample Input 2	Output for the Sample Input 2
15 0 0 1 4 3 2 6 0 6 3 6 5 9 1 10 4 12 5 13 0 13 5 15 2 18 3 19 4 20 1	7

Sample Input 3	Output for the Sample Input 3
17 10 4 12 5 19 4 6 5 20 1 0 0 1 4 13 0 6 0 6 3 18 3 3 2 9 1 13 5 15 2 20 2 15 3	8

Problem I
Quadratic Equation
Time Limit: 1 Second

For a given integer k ($\neq 0$), write a program to find all integers p such that the quadratic equation $x^2 + px + kp = 0$ has integer roots.

For example, when $k = 3$, all integers p that make the quadratic equation $x^2 + px + 3p$ have integer roots are listed in ascending order: $-4, 0, 12$, and 16.

Note that regardless of the value of k, if $p = 0$, the quadratic equation has an integer root of 0, so there always exists such p.

Input
Your program is to read from standard input. The first line contains an integer k ($-10^7 \leq k \leq 10^7;\ k \neq 0$).

Output
Your program is to write to standard output. Print exactly one line with two values separated with a single space character. The first one is the number of distinct integers p such that all the roots of the quadratic equation $x^2 + px + kp = 0$ are integers. The second value is the sum of such p's.

The following shows sample input and output for two cases.

Sample Input 1	Output for the Sample Input 1
3	4 24

Sample Input 2	Output for the Sample Input 2
-4	6 -48

Problem I
이차 방정식
제한 시간: 1 초

주어진 정수 k $(\neq 0)$에 대해, 이차 방정식 $x^2 + px + kp = 0$의 근이 모두 정수가 되도록 하는 서로 다른 정수 p를 모두 찾는 프로그램을 작성하시오.

예를 들어, $k = 3$인 경우 즉, $x^2 + px + 3p = 0$의 근이 모두 정수가 되도록 하는 모든 정수 p를 크기 순으로 나열하면 $-4, 0, 12, 16$이다.

참고로, k의 값이 어떻든 상관없이 $p = 0$이면 주어진 이차식은 정수근 0을 가지기 때문에 조건을 만족하는 정수 p는 하나 이상 존재한다.

Input
입력은 표준입력을 사용한다. 입력으로는 하나의 정수 k $(-10^7 \leq k \leq 10^7;\ k \neq 0)$가 주어진다.

Output
출력은 표준출력을 사용한다. 주어진 k에 대해 이차 방정식 $x^2 + px + kp = 0$의 근이 모두 정수가 되도록 하는 서로 다른 정수 p의 개수와 그 합을 출력하되 하나의 공백 문자로 두 값을 구분한다.

다음은 두 테스트 경우에 대한 입출력 예이다.

Sample Input 1	Output for the Sample Input 1
3	4 24

Sample Input 2	Output for the Sample Input 2
-4	6 -48

Problem J
Rescue Squad
Time Limit: 1 Second

As a town leader, you must form a rescue squad to help a neighboring town under attack by monsters. There are n knights in your town, numbered from 1 to n, and a knight i $(1 \le i \le n)$ has a positive integer level L_i. For efficient monster hunting, the knights who form the rescue squad must be able to trust one another. The trust relationship T between two knights i and j $(1 \le i < j \le n)$ is defined as follows:

$T(i,j) = 1$, if i and j trust each other;
$T(i,j) = 0$, if they do not.

The rescue squad S is a set of four distinct knights, and each knight in the squad must have a trust relationship $T = 1$ with at least two of the other three. The ability of S, $\text{Ability}(S)$, is defined as the sum of the levels of the four knights in S.

For example, suppose that $n = 5$ and the levels of the five knights are $L_1 = 3$, $L_2 = 2$, $L_3 = 3$, $L_4 = 7$, and $L_5 = 1$. If the trust relationship is defined as $T(1,2) = T(2,3) = T(3,4) = T(1,3) = T(1,4) = T(1,5) = 1$ and $T(2,4) = T(2,5) = T(3,5) = T(4,5) = 0$, then $S = \{1,2,3,4\}$ is the only possible rescue squad, and $\text{Ability}(S)$ is 15.

Given the levels of n knights and their trust relationships, write a program to find a squad S for which $\text{Ability}(S)$ is maximized and output its ability value.

Input
Your program is to read from standard input. The input starts with a line containing two integers, n and m $\left(4 \le n \le 1{,}000;\ 1 \le m \le \min\left(\frac{n(n-1)}{2},\ 25{,}000\right)\right)$, where n is the number of knights and m is the number of pairs of knights (i,j) such that $T(i,j) = 1$ and $i < j$. In the following n lines, the i-th line contains a positive integer that represents the level L_i $(1 \le L_i \le 10{,}000)$ of the knight i. In the following m lines, each line contains two integers, i and j $(1 \le i < j \le n)$ that represent a trust relationship $T(i,j) = 1$. There are no duplicate entries among the m lines describing the trust relationships, and for any pair of knights i' and j' that do not appear in the input, $T(i',j') = 0$.

Output
Your program is to write to standard output. Print exactly one line. The line should contain the ability $\text{Ability}(S)$ of a squad S with the maximum ability among the squads that can be formed. If no squad can be formed, print -1.

The following shows sample input and output for two test cases.

Sample Input 1	Output for the Sample Input 1
5 6 3 2 3 7 1 1 2 2 3 3 4 1 3 1 4 1 5	15

Sample Input 2	Output for the Sample Input 2
5 5 1 2 3 4 5 1 2 2 3 3 4 4 5 1 5	-1