The 42nd Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem Set

Please check that you have 12 problems that are spanned across 33 pages in total (including this cover page and Korean translated ones).

A. Closest Pair	(2 pages)	Korean translation available
B. Cycle Mean	(2 pages)	
C. Flow Graph Complexity	(2 pages)	Korean translation available
D. Grasshopper Route	(2 pages)	
E. Jerry and Tom	(2 pages)	
F. Leftmost Segment	(2 pages)	
G. Map Labeling	(2 pages)	Korean translation available
H. MultiMax	(1 page)	Korean translation available
I. Pizza Boxes	(2 pages)	Korean translation available
J. Pseudoknot	(2 pages)	
K. Registration	(1 page)	Korean translation available
L. Telescope	(2 pages)	

The memory limits for the twelve problems are all the same, 512MB.

The 42nd Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem A
Closest Pair
Time Limit: 1 Second

Given two sets P and Q of finitely many points in the plane, a closest pair of P and Q is a pair (p, q) of points $p \in P$ and $q \in Q$ such that the distance between p and q is the minimum among all pairs (p', q') with $p' \in P$ and $q' \in Q$.

Specifically, in this problem, by the *distance* between two points a and b in the plane, we mean:

$$d(a, b) = |x_a - x_b| + |y_a - y_b|$$

where x_a and y_a denote the x- and y-coordinates of point a, and x_b and y_b denote the x- and y-coordinates of point b. Then, a pair (p, q) with $p \in P$ and $q \in Q$ is a closest pair of P and Q if and only if the following holds:

$$d(p, q) = \min\{d(p', q') \mid p' \in P \text{ and } q' \in Q\}$$

Given two sets P and Q, write a program that computes the distance between a closest pair of P and Q and the number of distinct closest pairs of P and Q.

Note that you can assume the following on the input points in P and Q:

1. All the points in P lie on the horizontal line $y = c_1$ while all the points in Q lie on the horizontal line $y = c_2$ for some integers c_1 and c_2.
2. No two input points in P have the same coordinates; no two input points in Q have the same coordinates.

Input
Your program is to read from standard input. The input consists of four lines. The first line contains two integers, n ($1 \le n \le 500{,}000$) and m ($1 \le m \le 500{,}000$), where n is the number of points in set P and m is the number of points in set Q. In the second line, two integers c_1 and c_2 ($-10^8 \le c_1, c_2 \le 10^8$) are given in order, separated by a single space. In the third line, n distinct integers between -10^8 and 10^8, inclusively, are given, separated by a single space, that are the x-coordinates of the points in set P, while their y-coordinates are all the same as c_1. In the fourth line, m distinct integers between -10^8 and 10^8, inclusively, are given, separated by a single space, that are the x-coordinates of the points in set Q, while their y-coordinates are all the same as c_2.

Output
Your program is to write to standard output. Print exactly one line for the input. The line should contain two integers, separated by a single space, that represent the distance between a closest pair of P and Q and the number of closest pairs of P and Q in this order.

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

Sample Input 1	**Output for the Sample Input 1**
3 4 1 -3 3 0 6 -2 5 4 2	5 3

Sample Input 2	**Output for the Sample Input 2**
5 5 1 2 -4 -10 -2 0 -1 3 18 0 1 5	1 1

The 42nd Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem A
Closest Pair
Time Limit: 1 Second

평면 위의 유한개의 점을 포함하는 집합 P와 Q가 주어질 때, P와 Q사이의 최근접쌍(closest pair)이란 $p' \in P$와 $q' \in Q$를 만족하는 모든 쌍 (p', q') 중에서 거리가 가장 가까운 쌍을 말한다.

이 문제에서는 특히 임의의 두 점 a와 b 사이의 거리를 다음과 같은 함수로 정의하도록 한다.

$$d(a, b) = |x_a - x_b| + |y_a - y_b|$$

위 식에서 x_a와 y_a는 각각 점 a의 x-좌표 및 y-좌표를 의미하며, x_b와 y_b는 각각 점 b의 x-좌표 및 y-좌표를 의미한다. 이때, $p \in P$이고 $q \in Q$인 점의 쌍 (p, q)는 아래의 식을 만족하는 경우 두 집합 "P와 Q사이의 최근접쌍"이라고 부른다.

$$d(p, q) = \min\{d(p', q') \mid p' \in P \text{ and } q' \in Q\}$$

두 집합 P와 Q가 입력으로 주어질 때, P와 Q사이의 최근접쌍 간의 거리와 P와 Q사이의 서로 다른 최근접쌍의 개수를 계산하여 출력하는 프로그램을 작성하시오.

이 문제에서의 입력은 아래와 같은 조건을 만족한다.

1. 어떤 정수 c_1과 c_2에 대해, 집합 P에 속한 모든 점은 x 축에 평행한 직선 $y = c_1$ 위의 점들이며, 집합 Q에 속한 모든 점은 x 축에 평행한 직선 $y = c_2$ 위의 점들이다.
2. 집합 P에 속한 모든 점은 서로 다른 좌표를 갖는다. 또한, 집합 Q에 속한 모든 점은 서로 다른 좌표를 갖는다.

입력
프로그램의 입력은 표준 입력으로 받는다. 입력은 모두 네 줄로 이루어진다. 첫 줄에는 두 개의 정수 n $(1 \le n \le 500{,}000)$과 m $(1 \le m \le 500{,}000)$이 빈 칸을 사이에 두고 주어지며, 이는 각각 집합 P에 속한 점의 개수와 집합 Q에 속한 점의 개수를 의미한다. 둘째 줄에는 역시 두 개의 정수 c_1과 c_2 $(-10^8 \le c_1, c_2 \le 10^8)$가 빈 칸을 사이에 두고 주어진다. 셋째 줄에는 -10^8이상, 10^8이하인 n개의 서로 다른 정수가 주어지며, 이는 집합 P에 속한 점들의 x-좌표를 의미한다. 집합 P에 속한 점들의 y-좌표는 모두 c_1으로 같다. 넷째 줄에는 -10^8이상, 10^8이하인 m개의 서로 다른 정수가 주어지며, 이는 집합 Q에 속한 점들의 x-좌표를 의미한다. 집합 Q에 속한 점들의 y-좌표는 모두 c_2로 같다.

출력
표준 출력으로 답을 출력한다. 주어진 입력에 대해 P와 Q사이의 최근접쌍 간의 거리와 P와 Q사이의 최근접쌍의 개수를 두 개의 정수 형태로 빈칸을 사이에 두어 구분하여 한 줄에 출력한다.

다음은 두 개의 입력에 대한 출력 값을 나타낸 예제이다.

입력 예제 1	**입력 예제 1 에 대한 출력**
3 4 1 -3 3 0 6 -2 5 4 2	5 3

입력 예제 2	**입력 예제 2 에 대한 출력**
5 5 1 2 -4 -10 -2 0 -1 3 18 0 1 5	1 1

The 42nd Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem B
Cycle Mean
Time Limit: 2 Seconds

A metabolic network is modeled as a directed graph in which a vertex represents a state and an edge represents a transition from one state to another. Each edge has a weight like a cost or an energy that needs for the transition of the edge. The mean weight of a directed cycle in the metabolic network is the total weight of its edges divided by their number. In general, the efficiency of the network is measured by the minimum mean weight of directed cycles in the network. We now want to measure the efficiency of a given metabolic network by computing the minimum mean weight.

More precisely, a digraph (directed graph) $G = (V, E)$ of n vertices, associating each edge e with a positive weight, is given. A cycle C in G is simple if the vertices along C are all distinct. A weight $w(C)$ of a (directed) simple cycle C in G is the total weights of the edges in C, and the *mean weight* of C is $w(C)/|C|$, where $|C|$ is the number of edges of C or cycle length of C. The *minimum cycle mean* of G is the minimum mean weight of simple cycles in G. For simplicity, we assume that a digraph G is simple, that is, there is no self-loop edge from a vertex to itself, and there are no two or more edges from u to v for any distinct vertices u and v of G. Note that the length of any simple cycle in G is at least two.

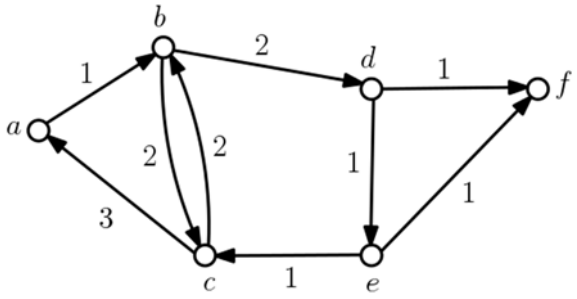


Figure B.1. A digraph $\boldsymbol{G}$ of 6 vertices and 9 edges.

For example, a digraph G in Figure B.1 has four directed simple cycles in total, $b \to c \to b$, $a \to b \to c \to a$, $b \to d \to e \to c \to b$, and $a \to b \to d \to e \to c \to a$, of cycle length 2, 3, 4, and 5, respectively. The weights of the cycles are 4, 6, 6, and 8, thus their means are $\frac{4}{2} = 2$, $\frac{6}{3} = 3$, $\frac{6}{4} = 1.5$, and $\frac{8}{5} = 1.6$, respectively. The cycle of the minimum mean is $b \to d \to e \to c \to b$ and its mean is 1.5.

You write a program to output the minimum cycle mean of a simple digraph G.

Input
Your program is to read from standard input. An input starts with a line containing two integers, n, m ($2 \leq n \leq 1{,}000, 1 \leq m \leq 10^5$), where n is the number of vertices and m is the number of edges in a digraph G. Vertices have distinct id numbers from 0 to $n-1$. In the following m lines, each of the m edges in G is given line by line. Each edge is represented by three integers u, v, w separated by a single space, where the edge is directed from u to v ($0 \leq u, v \leq n-1, u \neq v$) of weight w ($1 \leq w \leq 1{,}000$).

Output

Your program is to write to standard output. Print exactly one line for the input. The line should contain two integers a and b separated by a single space in this order such that a and b are relatively primes and $\frac{a}{b}$ is the minimum cycle mean of G. But if G has no cycle, you should output two zeroes separated by a single space.

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

Sample Input 1	Output for the Sample Input 1
6 9 0 1 1 1 2 2 2 1 2 2 0 3 4 2 1 1 3 2 3 4 1 4 5 1 3 5 1	3 2

Sample Input 2	Output for the Sample Input 2
13 15 0 2 1 2 1 1 1 0 2 5 1 1 5 2 1 5 6 1 3 4 1 8 9 2 9 8 1 6 8 1 6 7 2 10 7 1 11 7 1 10 11 1 7 5 2	4 3

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

The 42nd Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem C
Flow Graph Complexity
Time Limit: 1 Second

Soohwan is leading the International Committee for Program Complexity (ICPC), and the major task of ICPC is measuring the complexity of program codes. A well-known representation of a program is the control flow graph, where the nodes represent the program constructs and the edges the possible flow of controls. The cyclomatic complexity is a popular way to measure the complexity of a flow graph. For the digraph $G(V, E)$ representing a flow graph, the cyclomatic complexity $C(G)$ is defined by the formula $C(G) = |E| - |V| + 2$. For instance, for the flow graphs G_1 and G_2 in Figure C.1, the complexities of them are computed to be 3 ($C(G_1) = C(G_2) = 9 - 8 + 2 = 3$).

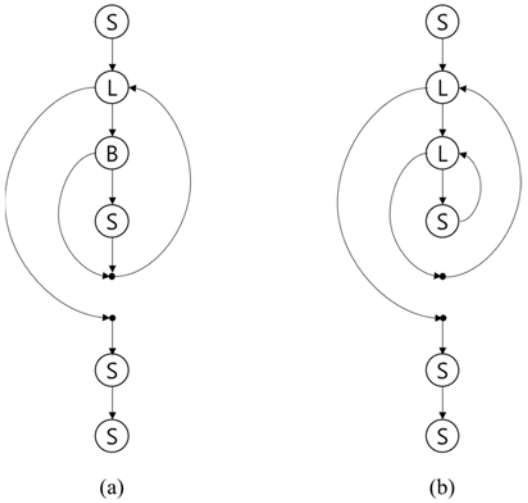


Figure C.1: Two flow graphs G_1 (a) and G_2 (b)

In Figure C.1, the labels B, L, and S denote the types of nodes: B for a branch, L for a loop, and S for a simple statement. The branch node B introduces additional forward edge skipping one or more statements. The loop node L, the target of an incoming backward edge, also introduces a forward edge exiting the loop. Note that the dots representing the closing points of loops or branches are nodes, too.

One critique for the cyclomatic complexity is that the backward edges introduced by loops are weighted equal to the forward edges. Therefore Soohwan devised a new measure imposing more weights on the backward edges than the forward ones. He classified the edges into two disjoint categories E_F for the forward and E_B for the backward ($E = E_F \cup E_B$ and $E_F \cap E_B = \emptyset$), and defined a new measure C_M, namely the flow graph complexity, using the formula $C_M(G) = |E_F| + W \times |E_B| - |V| + 2$ for a constant weight W for backward edges. Soohwan wants to validate this new measure.

You are to help Soohwan by writing a program calculating the new complexity for a given flow graph. For simplicity, only pretest loops (say, `while` loops in C) and only one-way branches (say, `if` statements without `else` clauses) are assumed. With this assumption, a flow graph can be represented by a string of node types defined as follows:

L for a loop node,
B for a branch node, and
S for a simple statement node.

L and B should be followed by the string representing its sub-structure enclosed by a pair of parentheses. Beware that the closing points are implicitly represented by parentheses. The sequence of node representations are separated by commas. According to this representation, the string representations of the graphs in Figure C.1 are as follows:

G_1: S,L(B(S)),S,S
G_2: S,L(L(S)),S,S

With the weight $W = 5$, the new complexities for them are $C_M(G_1) = |E_F| + W \times |E_B| - |V| + 2 = 8 + 5 \times 1 - 8 + 2 = 7$ and $C_M(G_2) = |E_F| + W \times |E_B| - |V| + 2 = 7 + 5 \times 2 - 8 + 2 = 11$.

Write a program to compute C_M given the weight W for backward edges and the string representing the flow graph.

Input

Your program is to read from standard input. The input starts with a line containing an integer, W ($1 < W \leq 27$), where W is the weight for the backward edge. In the following line, the string representation P for the flow graph is given. The length of P is less than 70,000. The string P consists of uppercase alphabets and punctuation symbols. To make the representation easy to read, P may contain spaces; the pairs of brackets [and] can be used instead of the pairs of parentheses and they should be matched. P may contain other punctuation symbols such as colon (:), semicolon (;), period (.) erroneously, which your program should detect them as invalid symbols. The input string P is invalid if (1) the parentheses and the brackets are unmatched, (2) it contains punctuation symbols other than parentheses, brackets, and commas, (3) the punctuation symbols are added or omitted wrongly such as S,,S, SS, or B,(S), and (4) it contains unknown types of node other than S, L, or B.

Output

Your program is to write to standard output. Print exactly one line for the input. The line should contain the flow graph complexity $C_M(G)$ for the given G represented by P. If input string P is invalid, print -1 instead.

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

Sample Input 1	Output for the Sample Input 1
5 S,L(L(S)),S,S	11

Sample Input 2	Output for the Sample Input 2
3 L(S, L[B(S),S])	8

Sample Input 3	Output for the Sample Input 3
11 B(S, L((F(S), S], S(S))	-1

The 42nd Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem C
Flow Graph Complexity
Time Limit: 1 Second

수환은 프로그램 복잡도 국제 위원회(ICPC: International Committee for Program Complexity)를 이끌고 있는데 위원회의 주요 임무는 프로그램 코드의 복잡도를 측정하는 것이다. 프로그램 표현법으로 제어 흐름 그래프(control flow graph)가 잘 알려져 있는데, 프로그램 구성 요소를 정점(node)으로 나타내고 가능한 제어 흐름을 간선(edge)으로 나타내는 방법이다. 순환복잡도(cyclomatic complexity)는 흐름 그래프의 복잡도 측정 방법으로 유명하다. 흐름 그래프를 나타내는 방향 그래프 $G(V,E)$에 대한 순환 복잡도 $C(G)$의 정의는 공식 $C(G) = |E| - |V| + 2$로 주어진다. 예컨대 그림 C.1의 그래프 G_1과 G_2에 대한 순환 복잡도는 3이다($C(G_1) = C(G_2) = 9 - 8 + 2 = 3$).

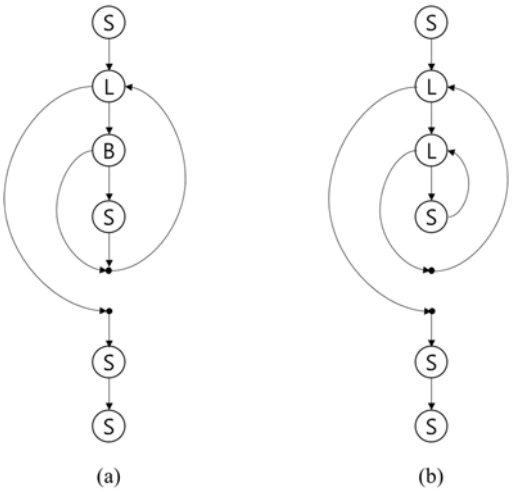


Figure C.1: 두 개의 흐름 그래프 (a) G_1과 (b) G_2

그림 C.1에서 레이블 B, L, S는 노드 종류를 나타내는데 B는 분기(branch), L은 반복(loop), S는 단순 문장을 나타낸다. 분기 노드 B에서는 하나 이상의 문장을 건너뛰는 순방향 간선(forward edge)이 하나 더 생성된다. 역방향 간선(backward edge)이 유입되는 반복 노드 L에서도 루프를 탈출하는 순방향 간선이 하나 더 생성된다. 반복과 분기가 끝나는 곳을 나타내는 점도 정점이라는 것에 주의하자.

순환 복잡도는 순방향 간선과 역방향 간선을 같은 비중으로 다룬다는 점에서 비판받고 있다. 그래서 수환은 순방향 간선보다 역방향 간선에 더 큰 가중치를 두는 새로운 복잡도를 고안하였다. 수환은 간선을 순방향 간선 E_F과 역방향 간선 E_B, 두 부류로 나누고($E = E_F \cup E_B$이고 $E_F \cap E_B = \emptyset$), 이른바 흐름 그래프 복잡도라는 새로운 측정 기준 C_M을 정의하였는데 이는 가중치 상수 W에 대하여 공식 $C_M(G) = |E_F| + W \times |E_B| - |V| + 2$으로 정의된다. 수환은 이 새로운 측정법의 타당성을 입증하려고 한다.

여러분은 주어진 흐름 그래프에 대해 새로운 복잡도를 계산하는 프로그램을 작성하여 수환을 도와주고자 한다. 문제를 간단히 하기 위해 조건을 먼저 검사하는 반복문(예컨대 C의 `while` 문)과 단방향 분

기문(예컨대 `else` 가 없는 `if` 문)만 허용된다고 가정하자. 이렇게 가정하면 흐름 그래프는 다음과 같은 노드 종류의 문자열로 나타낼 수 있다.

L: 반복문 노드
B: 분기문 노드
S: 단순 문장 노드

L과 B 다음에는 하부 구조를 나타내는 문자열을 괄호로 묶어 나타낸다. 반복문과 분기문의 끝을 나타내는 정점은 괄호를 통해 암묵적으로 표현된다는 점에 주의하자. 정점을 나타내는 문자열은 쉼표로 구분하여 나타낸다. 이 표기법에 따라 그림 C.1의 그래프를 문자열로 나타내면 다음과 같다.

G_1: S,L(B(S)),S,S
G_2: S,L(L(S)),S,S

가중치가 $W = 5$일 때 이들 그래프에 대한 새로운 복잡도는 $C_M(G_1) = |E_F| + W \times |E_B| - |V| + 2 = 8 + 5 \times 1 - 8 + 2 = 7$, $C_M(G_2) = |E_F| + W \times |E_B| - |V| + 2 = 7 + 5 \times 2 - 8 + 2 = 11$로 계산할 수 있다.

흐름 그래프 문자열과 역방향 간선 가중치 W가 주어졌을 때 C_M을 계산하는 프로그램을 작성하라.

입력
프로그램의 입력은 표준 입력으로 주어진다. 입력의 첫 줄에는 역방향 간선에 대한 가중치를 나타내는 정수 $W(1 < W \leq 27)$가 주어지고 그 다음 줄에는 흐름 그래프의 문자열 표기 P가 주어진다. P의 길이는 70,000이하이다. 문자열 P는 대문자 알파벳과 문장 부호로 구성된다. 읽기 편하도록 P에는 공백문자가 추가될 수 있고 괄호 대신 대괄호 [와]를 쓸 수 있는데 대괄호도 짝이 맞아야 한다. P에는 쌍점(:)이나 쌍반점(;), 마침표(.) 등이 실수로 포함될 수 있는데 프로그램은 이런 실수를 잘못된 입력으로 검출해야 한다. 입력 문자열 P는 다음과 같은 경우에 잘못된 입력인데, (1) 괄호나 대괄호가 맞지 않는 경우, (2) 괄호, 대괄호, 쉼표 외의 다른 문장 부호가 포함된 경우, (3) 문장 부호가 잘못 추가되거나 생략된 경우(예: S, , S나 SS, B, (S)), (4) S, L, B 외에 알려지지 않은 타입의 노드가 포함된 경우이다.

출력
프로그램의 출력은 표준 출력으로 출력한다. 입력에 대해 한 행만 출력하는데, 입력 문자열 P가 나타내는 그래프 G의 흐름 그래프 복잡도 $C_M(G)$를 출력한다. 문자열 P가 잘못되었다면 -1을 출력한다.

다음은 세 개의 입력에 대한 출력 값을 나타낸 예제이다.

입력 예제 1	**입력 예제 1 에 대한 출력**
5 S,L(L(S)),S,S	11

입력 예제 2	**입력 예제 2 에 대한 출력**
3 L(S, L[B(S),S])	8

입력 예제 3	**입력 예제 3 에 대한 출력**
11 B(S, L((F(S), S], S(S))	-1

The 42nd Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem D
Grasshopper Route
Time Limit: 1 Second

A grasshopper is planning to explore a tree, where a *tree* is an undirected graph in which any two vertices are connected by exactly one path. The grasshopper on a vertex of the tree can jump to, and only to, any other vertex within a distance of 3, where the *distance* between two vertices is the number of edges in the path connecting the two. The grasshopper wants to visit every vertex of the tree exactly once, starting at the vertex, s, it is currently on and ending at the vertex, t, it wants.

Given a pair of vertices s and t in a tree with n vertices, your job is to write an efficient program to report an ordering of the vertices of the tree, called a *grasshopper route*, according to which the grasshopper can accomplish what it wants. Specifically, a grasshopper route for s and t in the tree is an ordering $\langle u_1, u_2, \ldots, u_n \rangle$ of the vertices of the tree such that $u_1 = s$, $u_n = t$, and the grasshopper can jump and move from u_i to u_{i+1} for every $i \in \{1, 2, \ldots, n-1\}$. Fortunately, it was proven early in 1960 that each pair of vertices of a tree are joined by a grasshopper route.

In the tree shown in Figure D.1 below, for example, there is a grasshopper route $\langle 7,6,5,4,1,2,3,8,9,11,12,10 \rangle$ for $s = 7$ and $t = 10$. The grasshopper can jump from vertex 5 to vertex 4 because the distance between the two is at most 3; however, it cannot jump from vertex 5 to vertex 3. As you guessed, there may exist more than one grasshopper routes.

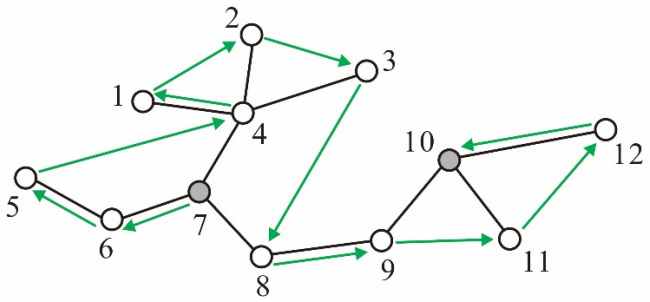


Figure D.1: A grasshopper route from s to t, where $s = 7$ and $t = 10$.

Input
Your program is to read from standard input. The first line contains an integer, n, representing the number of vertices of the input tree, where $2 \leq n \leq 100{,}000$. It is followed by $n-1$ lines, each contains two positive integers u and v that represent an edge between vertex u and vertex v of the input tree. It is assumed that the vertices are indexed from 1 to n. The last line contains two distinct integers s and t, where $s, t \in \{1, \ldots, n\}$, that respectively represent the start and end vertices of a grasshopper route.

Output
Your program is to write to standard output. Print out a required grasshopper route in n lines, containing, one by one, the vertices encountered when we traverse the route from s to t.

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

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

Sample Input 2	Output for the Sample Input 2
12 4 1 4 2 4 3 4 7 5 6 6 7 7 8 8 9 9 10 11 10 12 10 7 10	7 6 5 4 1 2 3 8 9 11 12 10

The 42nd Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem E
Jerry and Tom
Time Limit: 1 Second

Naughty mouse Jerry and his friend mice sometimes visit a vacant house to play the famous children game 'hide and seek' and also to adjust the length of their teeth by gnawing furniture and chairs left there. If we look down the house from the sky the boundary of it composes an orthogonal polygon parallel to the xy-axes as shown in the figure below. In other words, every wall of the house is either horizontal or vertical.

Tom, a threatening cat to them, sometimes appears in the house while they are enjoying the game. In that case Jerry and his friends should hide into the rat's holes at the bottom on the walls. There are two rules which must be held for them to hide into the holes:

1. Each hole can afford at most k mice.
2. Each mouse can enter the hole which can be seen by it. In other words, a mouse cannot enter the hole which is hidden by any wall. (That is, if the connecting line between a mouse and a hole intersects either any wall or any corner point of the house, the hole is considered hidden from the mouse.)

For example, consider a situation where three mice and three holes are in the house as shown in Figure E.1. Each circle on the boundary denotes a hole. Assuming that $k = 1$, i.e., only one mouse is allowed to hide into each hole, with the situation shown in the left figure, when Tom appears all the three mice can hide. But for the case shown in the right figure it is impossible for all the mice to hide.

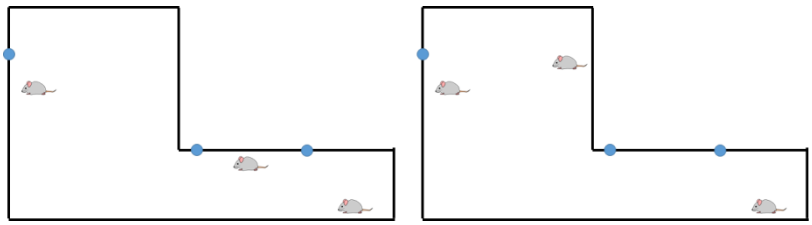


Figure E.1: Illustration to show two situations: 1. All mice can hide (left) and 2. They cannot (right)

You can assume:

1. Every mouse is strictly inside the house, which means that no mouse is on the wall
2. Every hole is on the wall.
3. No two holes locate at the same spot.
4. No two mice locate at the same spot.

Given a situation explained above, you are to write a program which determines whether all the mice can hide or not.

Input

Your program is to read from standard input. The input starts with a line containing four integers, $n,\ k,\ h$, and m, where $n(1 \leq n \leq 1{,}000)$ is the number of the corner points of a house, $k(1 \leq k \leq 5)$ the maximum

number of mice each hole can afford, $h(1 \leq h \leq 50)$ the number of holes, $m(1 \leq m \leq k \cdot h)$ the number of mice. In each of the following n lines, each coordinate of the corner points of the house is given in counter clockwise order. Each point is represented by two integers separated by a single space, which are the x-coordinate and the y-coordinate of the point, respectively. Each coordinate is given as an integer between -10^9 and 10^9, inclusively. In each of the following h lines, two integers x and y are given, which represent the coordinate (x, y) of each hole. In each of the following m lines, two integers x and y are given, which represent the coordinate (x, y) of each mouse.

Output

Your program is to write to standard output. Print exactly one line for the input. Print `Possible` if all the mice can hide into the rat's holes holding the constraints explained above. Otherwise print `Impossible`.

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

Sample Input 1	Output for the Sample Input 1
6 1 3 3 0 0 100 0 100 50 40 50 40 70 0 70 0 55 55 50 80 50 15 65 90 10 92 10	Possible

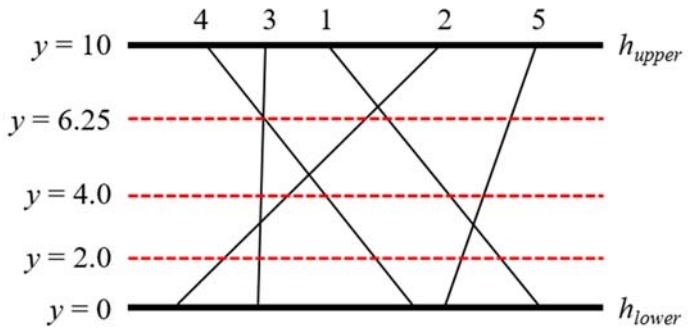
Sample Input 2	Output for the Sample Input 2
6 1 3 3 0 0 100 0 100 50 40 50 40 70 0 70 0 55 55 50 80 50 15 65 90 10 30 66	Impossible

The 42nd Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem F
Leftmost Segment
Time Limit: 1 Second

There are two distinct horizontal lines h_{upper} and h_{lower} in the xy-coordinate plane and n line segments connecting them. One endpoint of each segment lies on h_{upper} and the other on h_{lower}. All endpoints of the segments are distinct. All segments are numbered from 1 to n. Consider a horizontal line h_i located between h_{upper} and h_{lower}, which will be given by a query. The line h_i crosses all segments definitely. We want to know which segment intersects at the leftmost with h_i. You can observe that the leftmost intersection point between the segments and a query line may lie on one or more segments since two or more segments may intersect at a single point. In that case, the leftmost segment is defined as the segment which has the leftmost endpoint on h_{upper}.

For example, 5 segments and 3 query lines are given in the plane as shown in the figure below. The leftmost segment that intersects with a query line of $y = 2.0$ is 2 and the leftmost segment that intersects with a query line of $y = 4.0$ is 3. The query line of $y = 6.25$ crosses the intersection point between the segments 3 and 4, hence the leftmost segment is 4 by definition.



Given n segments connecting two horizontal lines and m queries, you are to write a program to find the leftmost segment that intersects with each query line.

Note that two or more segments may intersect at a single point. You should be also careful of round-off errors caused by the computer representation of real numbers.

Input
Your program is to read from standard input. The input starts with a line containing two integers, $maxY$ and $minY$ ($-1{,}000 \le minY < maxY \le 1{,}000$), where $maxY$ and $minY$ represent the y-coordinates of the upper horizontal line and the lower horizontal line, respectively. The next line contains an integer n ($1 \le n \le 100{,}000$) which is the number of segments connecting two horizontal lines. All segments are numbered from 1 to n in order given as the input. In the following n lines, each line contains two integers $upperX$ and $lowX$ ($-500{,}000 \le upperX, lowX \le 500{,}000$) which represent the x-coordinates of the upper endpoint and the lower endpoint of a line segment, respectively. All endpoints are distinct. The next line contains an integer m ($1 \le m \le 100{,}000$) which is the number of queries. In the following m lines, each line

contains a y-coordinate given for the query horizontal line, which is a real number between $minY$ and $maxY$ exclusive and the number of digits after the decimal point is 1 or more and 3 or less.

Output

Your program is to write to standard output. Print exactly one line for each query in order given as the input. The line should contain the leftmost segment number which intersects with the query horizontal line.

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

Sample Input 1

```
10 0
5
5 12
9 0
4 3
1 8
12 9
3
6.25
4.0
2.0
```

Output for the Sample Input 1

```
4
3
2
```

Sample Input 2

```
5 -5
4
0 0
1 1
2 2
3 4
3
0.0
2.05
-3.275
```

Output for the Sample Input 2

```
1
1
1
```

Sample Input 3

```
3 0
3
6 7
8 5
2 11
3
1.0
1.499
1.501
```

Output for the Sample Input 3

```
2
2
3
```

Problem G
Map Labeling
Time Limit: 1 Second

Map labeling is to place extra information, usually in the form of textual labels, next to features of interest within a map. The typical features depicted on a map are line features(e.g. roads, rivers, etc.), area features(countries, forests, lakes, etc.), and point features(villages, cities, etc.). In this problem, only the point features will be considered.

The basic requirements on the map labeling are that the labels do not overlap with each other and that they are close to the features they are associated with. However, this is not always possible to be achieved, for example, in the case where the labels are too large or the feature set is too dense. In this problem, the labels may be far from their features such that they are pairwise disjoint. But each feature should be connected to its associated label through a polygonal line including a straight line, called a *connector*. Clearly, the connectors should not intersect with each other. The connectors are only of two kinds. One consists of a single vertical line, called a *straight connector* and the other consists of three connected line segments, that is, vertical, horizontal, and vertical segments, called a *bended connector*. See Figure G.1.

Specifically, there is a straight line L, considered to be the x-axis, on which n points, corresponding to the point features, lie. The locations of the n points are strictly different. In this problem, the labels are considered as rectangular areas with height 1 on the plane. So each point p_i is associated with an axis-parallel rectangular label l_i of width w_i and height 1. Note that the heights of all the labels are identical. These rectangular labels should be pairwise disjoint, but the boundaries of two labels can be touched. Consider a line U which is parallel to L, above L, and has a vertical distance 1 from L. The labels l_i should be placed such that their lower sides are attached on U and they are above U as shown in Figure G.1.

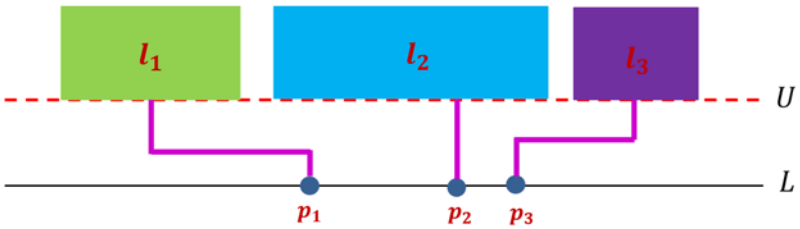


Figure G.1 Rectangular labels of point features

You write a program to find the placements of labels such that the number of bended connectors is minimized. For example, for the point features and the labels given in Figure G.1, the placement of labels shown in Figure G.2 minimizes the number of bended connectors.

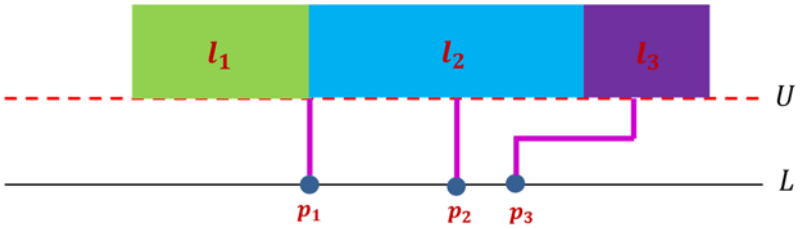


Figure G.2 Optimal placement of labels

Input

Your program is to read from standard input. The input starts with a line containing an integer, n ($1 \leq n \leq 10{,}000$), where n is the number of points on the line L, corresponding to the point features. In the i^{th} line of the following n lines, the coordinate a_i of the i^{th} point of the n points is given, where a_i is an integer and $0 \leq a_i \leq 10^8$. The coordinates of the n points are strictly different, that is, $a_i \neq a_j$ if $i \neq j$. In the i^{th} line of the following n lines, the width w_i of the associated label of the i^{th} point is given, where w_i is an integer and $1 \leq w_i \leq 10^5$. Note that the heights of all the labels are 1.

Output

Your program is to write to standard output. Print exactly one line for the input. The line should contain the minimum number of bended connectors among all the valid placements of labels.

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

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

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

The 42nd Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem G
Map Labeling
Time Limit: 1 Second

지도에서 흥미로운 지형지물(feature) 옆에 레이블(대개 문자 레이블) 형태로 부가적인 정보를 추가하는 것을 지도 레이블링(map labeling)이라고 한다. 지도의 전형적인 지형지물들로는 선(예를 들어, 도로, 강 등), 영역(나라, 숲, 호수 등), 점(마을, 도시 등) 형태가 있다. 본 문제에서 다루는 지형지물은 점 형태이다.

지도 레이블링의 기본 요구사항은 레이블들이 서로 겹치지 말아야 한다는 것과 레이블이 관련 지형지물들에 가까이 위치해야 한다는 것이다. 하지만 이 조건이 항상 만족되는 것은 아니다. 예를 들어서 레이블들이 아주 크거나 지형지물들이 너무 밀집된 경우에는 이 조건을 만족할 수 없다. 본 문제에서는 레이블들이 서로 겹치지 않는다면 관련 지형지물로부터 떨어져 있을 수 있는 경우를 다룰 것이다. 그러나 각 지형지물은 (직선을 포함한) 꺾은선(polygonal line)에 의해서 관련 레이블에 연결되어야 한다. 이 꺾은선을 *연결선(connector)*이라고 부른다. 분명 연결선들은 서로 교차하면 안 된다. 연결선들은 단 두 종류뿐인데, 하나는 한 개의 수직선으로 구성된 것으로 *직선 연결선(straight connector)*이라고 부린다. 다른 하나는 세 개의 연결된 선분으로 구성된 것으로 *꺾인 연결선(bended connector)*이라고 부른다, 여기서, 꺾인 연결선은 수직, 수평, 그리고 수직 선분으로 구성된 형태이다. 그림 G.1 을 보자.

구체적으로, 점 지형지물에 대응하는 n 개의 점들이 놓여 있는 직선 L 이 있는데 이 직선을 x-축으로 간주할 수 있다. n 개 점들의 위치는 모두 서로 다르다. 본 문제에서 레이블은 평면 상에 높이 1 인 사각형 영역이다. 즉, 각 점 p_i 의 레이블 l_i는 너비 w_i와 높이 1인 축에 평행한 사각형이다. 모든 레이블의 높이가 같음에 주목하자. 이 사각형 레이블들은 서로 겹치지 말아야 하지만 두 레이블의 경계선은 붙어도 된다. 직선 L에 평행하고 L 위로 수직 거리 1에 위치한 직선 U를 생각하자. 그림 G.1 처럼 레이블 l_i는 아래 변이 U와 만나며 U 위에 놓여야 한다.

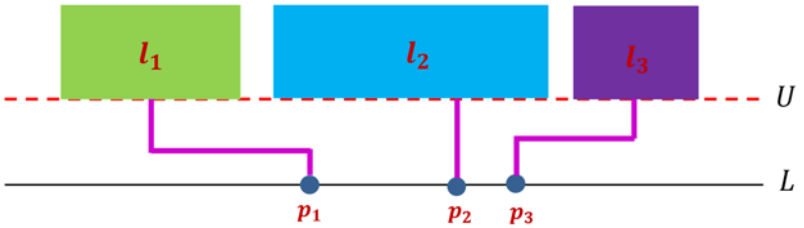


Figure G.1: 점 지형지물의 사각형 레이블

여러분은 꺾인 연결선의 수가 최소가 되도록 레이블들의 위치를 결정하는 프로그램을 작성해야 한다. 예를 들어, 그림 G.1 의 레이블들을 그림 G.2 처럼 놓으면 꺾인 연결선의 수가 최소가 된다.

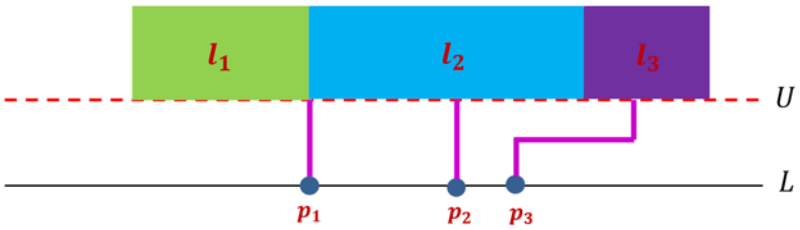


Figure G.2: 레이블의 최적 위치

입력

프로그램 입력은 표준 입력으로 주어진다. 입력은 정수 n ($1 \le n \le 10{,}000$)을 포함하는 줄로 시작한다. 여기서, n은 점 지형지물들에 대응하는 직선 L 상의 점들의 수이다. 다음 이어지는 n 개 줄의 i번째 줄에는 n 개의 점들 중에 i번째 점의 좌표 a_i가 주어진다. 여기서, a_i 는 정수이고 $0 \le a_i \le 10^8$. n 개 점들의 좌표는 모두 서로 다르다, 다시 말해, $i \ne j$ 이면, $a_i \ne a_j$. 다음 이어지는 n 개 줄의 i번째 줄에는 i번째 점의 관련 레이블의 너비 w_i가 주어진다. 여기서, w_i 는 정수이고 $1 \le w_i \le 10^5$. 모든 레이블들의 높이가 1 임을 상기하자.

Output

프로그램 출력은 표준 출력으로 출력한다. 입력에 대해 정확히 한 줄에 출력한다. 이 줄에는 가능한 레이블링 중에서 꺾인 연결선의 최소 수를 출력한다.

다음은 두 테스트 케이스에 대한 샘플 입력과 출력을 보여준다.

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

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

The 42nd Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem H
MultiMax
Time Limit: 1 Second

There are n cards, each with an integer on it where two or more cards can have the same integer. From these cards, we want to select two or three cards such that the product of numbers on the selected cards is maximum. For example, assume that there are 6 cards with numbers: $5, 10, -2, 3, 5$, and 2. If we select three cards with numbers, $5, 10$, and 5, then the product of the three numbers is 250, which is the maximum product. Let us consider another example of four cards with numbers: $10, 0, -5, 2$. In this example, if we select two cards with numbers, 10 and 2, then the product of these two numbers is 20, which is the maximum product.

Given n numbers on cards, write a program to compute the maximum product of numbers on two or three cards.

Input
Your program is to read from standard input. The input starts with a line containing an integer, n ($3 \leq n \leq 10{,}000$), where n is the number of cards. In the following line, n numbers on the cards are given. These numbers are integers between $-1{,}000$ and $1{,}000$ inclusively.

Output
Your program is to write to standard output. Print exactly one line which contains the maximum product .

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

Sample Input 1	Output for the Sample Input 1
6 5 10 -2 3 5 2	250

Sample Input 2	Output for the Sample Input 2
4 10 0 -5 2	20

The 42nd Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem H
MultiMax
제한 시간: 1 초

각 카드에 정수가 쓰여 있는 n장의 카드가 있다. 같은 수가 쓰여 있는 카드가 여러 장 있을 수 있다. 이들 n장의 카드에서 두 장 혹은 세 장의 카드를 선택한다. 이때, 선택 가능한 두 장의 카드에 있는 수들의 곱과 세 장 카드에 있는 수들의 곱 중에서 최대값을 구하고자 한다. 예를 들어 $n = 6$이고, 카드에 쓰여 있는 수들이 $5, 10, -2, 3, 5, 2$라 하자. 쓰여 있는 수가 $5, 10, 5$인 세 장의 카드를 선택하면 이들 수의 곱은 250이고, 이는 구하고자 하는 최대값이다. 다른 예로, $n = 4$이고, 카드에 쓰여 있는 수들이 $10, 0, -5, 2$라 하자. 쓰여 있는 수가 $10, 2$인 두 장의 카드를 선택하면 두 수 곱은 20이고, 이는 구하고자 하는 최대값이다.

정수가 쓰여 있는 n장 카드가 주어질 때, 두 장 카드에 있는 수들의 곱과 세 장 카드에 있는 수들의 곱 중에서 최대값을 구하는 프로그램을 작성하시오.

Input
입력은 표준입력을 사용한다. 첫 번째 줄에 카드 개수를 나타내는 양의 정수 n $(3 \le n \le 10{,}000)$이 주어진다. 다음 줄에 카드에 있는 n 개의 수들이 주어진다. 이들 수는 $-1{,}000$ 이상 $1{,}000$이하의 정수이다.

Output
출력은 표준출력을 사용한다. 두 수 곱과 세 수 곱 중에서 최대값을 한 줄에 출력한다.

다음은 두 테스트경우에 대한 입출력 예이다.

Sample Input 1	Output for the Sample Input 1
6 5 10 -2 3 5 2	250

Sample Input 2	Output for the Sample Input 2
4 10 0 -5 2	20

The 42nd Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem I
Pizza Boxes
Time Limit: 1 Second

There are pizza boxes all of which have the same dimensions. The boxes are stacked in piles, forming a three-dimensional grid where the heights are all different. The view from front shows the height of the tallest pile in each column, the view from the side shows the height of the tallest pile in each row.

What is the maximum number of pizza boxes we can remove without changing the front and side views? In the following example, Figure I.2 shows the solution of Figure I.1(a) case. In Figure I.1(a) and Figure I.2, each number (height) represents the number of boxes stacked.

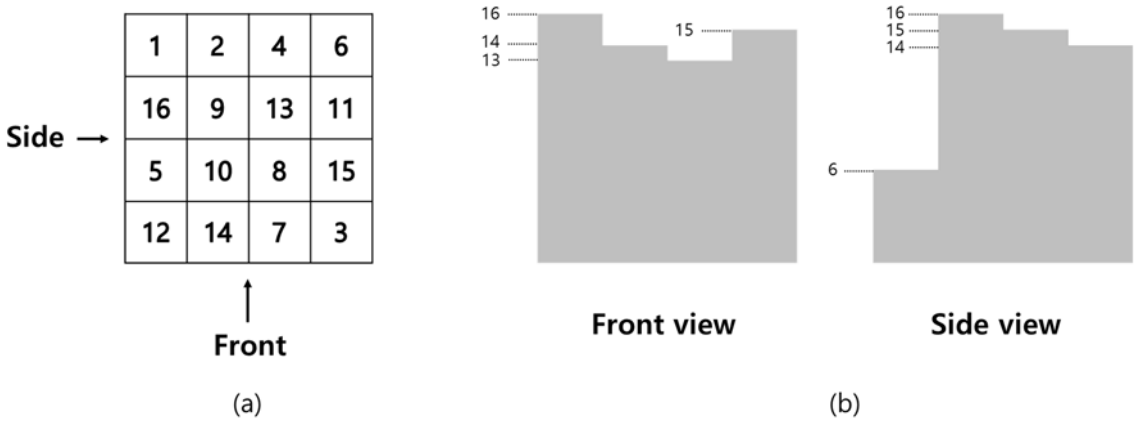


Figure I.1. (a) Grid of heights and (b) the corresponding views.

0	0	0	6
16	0	13	0
0	0	0	15
0	14	0	0

Figure I.2. Grid of heights after removing boxes.

Your task is to compute the maximum number of pizza boxes that can be removed without changing the original front and side views.

Input
Your program is to read from standard input. The input contains two integers, n and m ($1 \leq n, m \leq 1{,}000$), the number of rows and columns in the grid, respectively. Each of the following n lines contain m integers,

the number of pizza boxes (heights) in the corresponding row. All heights are between 0 and 10^9 inclusive and the heights are all different.

Output

Your program is to write to standard output. Print exactly one line for the input. The line should contain the maximum number of pizza boxes that can be removed without changing the original views.

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

Sample Input 1	Output for the Sample Input 1
4 4 1 2 4 6 16 9 13 11 5 10 8 15 12 14 7 3	72

Sample Input 2	Output for the Sample Input 2
3 5 1 11 25 20 23 17 2 16 21 15 10 3 12 24 22	101

The 42nd Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem I
Pizza Boxes
Time Limit: 1 Second

같은 크기의 피자 상자들이 있다. 이 상자들이 삼차원 격자모양으로 쌓여져 있다. 단, 각 격자마다 쌓여있는 상자들의 수(높이)는 모두 다르다. 이 상자 더미를 보면 쌓여있는 상자들의 윤곽선만 보이는데, 정면에서는 각 열에서 가장 높이 쌓여져 있는 격자의 상자들에 의해서 윤곽선이 결정되고, 측면에서 보면 각 행에서 가장 높이 쌓여져 있는 격자의 상자들에 의해 윤곽선이 결정된다.

정면에서 보는 상자들의 윤곽선과 측면에서 보는 상자들의 윤곽선이 변하지않도록 하면서 상자들의 일부를 제거한다고 할 때, 제거할 수 있는 상자의 최대 개수는 몇개인가? 아래의 예에서, 그림 I.2 는 그림 I.1(a)의 경우에서 정면과 측면에서 본 상자들의 윤곽선에 변화가 없도록 상자를 제거한 결과를 보여준다. 그림 I.1(a)와 그림 I.2 의 각 숫자는 쌓여져 있는 상자의 수(높이)를 나타낸다.

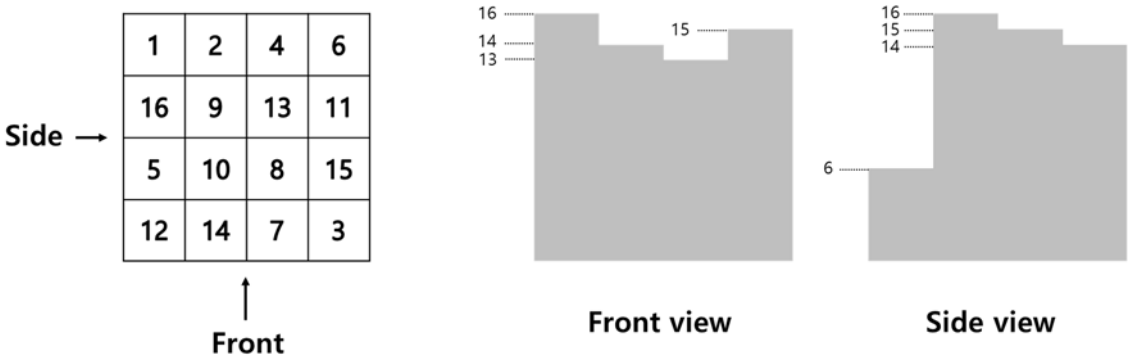


그림 I.1. (a) 상자들의 높이, (b) 정면 및 측면에서 보이는 상자들의 윤곽선.

0	0	0	6
16	0	13	0
0	0	0	15
0	14	0	0

그림 I.2. 윤곽선의 변화없이 상자들을 제거한 후의 상자들의 높이.

정면과 측면에서 보이는 상자들의 높이가 변함이 없이 최대로 제거할 수 있는 상자들의 수를 구하는 프로그램을 작성하라.

Input

입력 데이터는 표준입력을 사용한다. 입력의 첫번째 줄에는 상자들이 놓여있는 격자의 행과 열의 크기를 나타내는 두 정수 n 과 m $(1 \leq n, m \leq 1{,}000)$이 주어진다. 두번째 줄부터 n개의 줄에는 각각의 열에 쌓여있는 상자들의 수(높이)를 나타내는 m개의 정수가 주어진다. 상자들의 수(높이)는 0 이상 10^9이하의 정수이며, 쌓여있는 상자들의 수는 모두 다르다.

Output

출력은 표준출력을 사용한다. 정면과 측면에서 본 상자들의 윤곽선이 변하지않도록 하면서 상자들을 제거한다고 할 때, 제거할 수 있는 상자의 최대 개수를 한 줄에 출력한다.

다음은 두개의 테스트 데이터에 대한 입력과 출력의 예이다.

Sample Input 1	**Output for the Sample Input 1**
4 4 1 2 4 6 16 9 13 11 5 10 8 15 12 14 7 3	72

Sample Input 2	**Output for the Sample Input 2**
3 5 1 11 25 20 23 17 2 16 21 15 10 3 12 24 22	101

The 42nd Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem J
Pseudoknot
Time Limit: 2 Seconds

You work at a biological company and are currently developing a new type of vaccine for flu. A crucial step for making a new vaccine is to analyze an RNA strand and find a big chunk of the RNA strand that can be linked together. The RNA structure plays an important role in cells and molecular evolution. Thus, it is very important to calculate an RNA structure with a maximum number of links, which would use the optimal thermodynamic energy. This problem is called the pseudoknot prediction problem. When we regard an RNA strand as a string w, a pseudoknot string is in the form of $w = uvz^R u^R yz$, where u, v, z, y are substrings of w, $|u|, |z| > 0$ and $|v|, |y| \geq 0$. Here u^R and z^R denote the reversal of u and z, respectively, and then (u, u^R) and (z, z^R) form two links.

Figure J.1. An example of pseudoknot and non-pseudoknot

For example, the string in Figure J.1(a) is a pseudoknot since we can decompose the string w into six substrings u, v, z^R, u^R, y, z such that $u =$ ICPC, $v =$ IN, $y =$ FOR and $z =$ KOREA. Note that the two bold substrings (u, u^R) are reversal to each other, and the two underlined substrings (z, z^R) are reversal to each other. Now consider the string in Figure J.1(b). No matter how we decompose the string, we cannot make the pseudoknot structure. Therefore, we say that the string is not a pseudoknot.

Figure J.2. Another pseudoknot example

The string in Figure J.2. is also a pseudoknot. Note that here v, y are the null string (i.e., $|v| = |y| = 0$). We only require that u, z should be non-null strings ($|u|, |z| > 0$) and v, y can be a null-string in the pseudoknot structure.

Figure J.3. An example of two different pseudoknot structures for a same string

For some strings, there might be more than one possible pseudoknot depending on how we decompose the string. For instance, we have two different pseudoknot structures for the same string in Figure J.3. In biology, it is desirable to maximize the number of links, which is the reversal matching in pseudoknot. This implies that Figure J.3.(a) is more desirable than Figure J.3.(b) and, thus, it is better to report (a) instead of (b) for the input string in Figure J.3.

Input

Your program is to read from standard input. The input is a single string from the alphabet of uppercase letters without whitespace and line-break. The length of the string is between 1 and 200,000.

Output

Your program is to write to standard output. Print exactly one line for the input. The line should contain an integer for the largest t such that the input string w is a pseudoknot under the required conditions, $w = uvz^R u^R yz$, $|u|, |z| \geq t$ and $|v|, |y| \geq 0$. If no such t exists (i.e., w is not a pseudoknot), then your program should output -1.

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

Sample Input 1	Output for the Sample Input 1
ASLSDAAAFFSFAFDSLSAASFASDDFAFS	4

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

Sample Input 3	Output for the Sample Input 3
ICPCAEROKCPCIKOREA	4

The 42nd Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem K
Registration
Time Limit: 1 Second

Print out your team's DOMJudge account ID and password.

Input
No input is given for this problem.

Output
Your program is to write to standard output. Print exactly two lines. The first line should contain your DOMJudge account ID, and the second line should contain your DOMJudge account password.

The following shows sample input and output, where the account ID is `team123` and the password is `passw0rd`. Notice that no input is given.

Sample Input	Output for the Sample Input
	`team123` `passw0rd`

The 42nd Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem K
등록
Time Limit: 1 Second

자신이 속해있는 팀의 DOMJudge ID 와 패스워드를 그대로 출력하는 프로그램을 작성하시오.

Input
이 문제는 입력이 없다.

Output
표준출력(standard output)으로 출력해야 한다. 첫 줄에 Domjudge ID, 둘째 줄에 패스워드를 출력한다.

다음은 ID 가 `team123` 번, 패스워드가 `passw0rd` 인 경우의 입출력 예제이다. 참고로 입력이 없는 것에 주의한다.

Sample Input	Output for the Sample Input
	`team123` `passw0rd`

The 42nd Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem L
Telescope
Time Limit: 1 Second

A company named ICPC (Interstellar Cosmology Progress Corporation) is working on a new telescope. It receives optical signals with an $m \times l$ grid of sensors. Each sensor detects the color and the intensity of light it receives. Then it filters noises and multiplies the intensity of light by some constant. All the lights sensors received are then merged on a rectangular film. ICPC believes that this telescope can easily beat current optical or radar telescopes because traditional noises are filtered.

However, an engineer found a critical flaw in design. The film can withstand the light with intensity up to W. If it is exposed to the light with intensity greater than W, the telescope may break down.

Consider what will happen in practice.

1. The telescope is on one spot in the sky and as time goes on, it moves to the right. Or, it scans the sky from left to right.
2. The telescope is considered as an $m \times l$ grid. The sensor at cell (i, j) of the telescope $(1 \le i \le m, 1 \le j \le l)$ amplifies the light it receives by $P(i, j)$ where $0 \le P(i, j) \le 100$.
3. The path where the telescope scans in the sky is an $m \times n$ grid. $T(i, j)$ is the intensity of light at cell (i, j) of the path $(1 \le i \le m, 1 \le j \le n)$.
4. Initially, the telescope is at position 1 of the path: that is, the sensor at cell (i, j) $(1 \le i \le m, 1 \le j \le l)$ of the telescope is aligned with cell (i, j) of the path. Note that one cell of the path is covered by exactly one cell of the telescope.
5. In k minutes, the telescope is at position $k + 1$: the sensor at cell (i, j) of the telescope is aligned with cell $(i, j + k)$ of the path.
6. The intensity of light at position $k + 1$ is $W_k = \sum_{i=1}^{m} \sum_{j=1}^{l} T(i, j + k)P(i, j)$. If $W_k > W$, then the telescope may break down. Note that the last position of the telescope is $n - l + 1$.

Consider the following example. The path is a 3×5 grid, the telescope is a 3×3 grid, and $W = 20$. Each $T(i, j)$ is written on the left upper side of the grid and each $P(i, j)$ is written on the lower right side of the grid. At position 1, the intensity of light is $1 \times 1 + 4 \times 1 + 11 \times 1 = 16 < W$. At position 2, it is $3 \times 1 + 3 \times 1 + 3 \times 1 = 9 < W$. You can see that it is smaller than W at position 3 too.

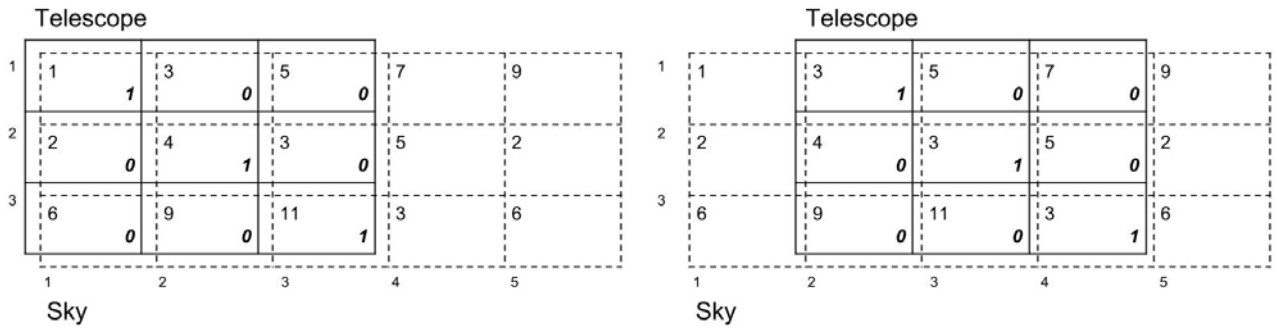


Figure L.1: An example of a telescope moving the path from left to right. (left) The telescope is at position 1 of the path. (right) Now the telescope moved to position 2.

Now you want to know how many times the telescope receives light with intensity greater than W, given the information on the sky and the telescope.

Input

Your program is to read from standard input. The input starts with a line containing three integers, n ($l \leq n \leq 10{,}000$), l ($2 \leq l \leq 3{,}000$), m ($2 \leq m \leq 100$), and W ($0 \leq W \leq 10^4 lm$), where n is the number of columns in the path, l is the number of columns in the telescope, m is that of rows, and W is the threshold defined above. In the following m lines, information on each row of the path is given line by line. That is, $T(i,j)$ is the jth number of line $i+1$. Also, $0 \leq T(i,j) \leq 100$. In the following m lines, information on each row of the sensors is given line by line. That is, $P(i,j)$ is the jth number of line $i+m+1$.

Output

Your program is to write to standard output. Print exactly one line for the input. The line should contain the number of times when the intensity of light on the film is greater than W.

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

Sample Input 1	Output for the Sample Input 1
4 2 2 40 1 2 3 4 1 0 0 0 10 10 10 10	2

Sample Input 2	Output for the Sample Input 2
5 3 2 20 1 3 5 7 9 2 4 3 5 2 1 0 0 0 1 0	0