The 41st 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 24 pages in total (including this cover page).

A. Binary Tree	(2 pages)	Korean translation available
B. Diameter	(2 pages)	
C. Frogs	(3 pages)	Korean translation available
D. Message Passing	(2 pages)	
E. Meteor Shower	(2 pages)	
F. Palindromic	(1 page)	
G. Planar Drawing	(3 pages)	
H. Project Team	(2 pages)	
I. Q-Index	(1 page)	Korean translation available
J. Railway	(2 pages)	Korean translation available
K. Registration	(1 page)	Korean translation available
L. Trucks	(2 pages)	Korean translation available

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

Problem A

Binary Tree

Time Limit: 1 Second

We are given a full binary tree of height k, where each edge has a positive weight. The full binary tree of height k has $(2^{k+1} - 1)$ nodes containing 2^k leaves. The distance from the root to a leaf is the sum of weights of all the edges on the path from the root to the leaf. In this problem, we would like to increase the weights of certain edges so that all root-to-leaf paths have the same distance and keep the sum of all the edge weights as small as possible.

For example, consider a full binary tree of height 2 in Figure 1(a). A number alongside an edge indicates the weight of the edge. The solution of this instance is shown in Figure 1(b). That is, all root-to-leaf paths have the same distance 5, and the sum of all the edge weights is 15 which is the minimum possible value in this case.

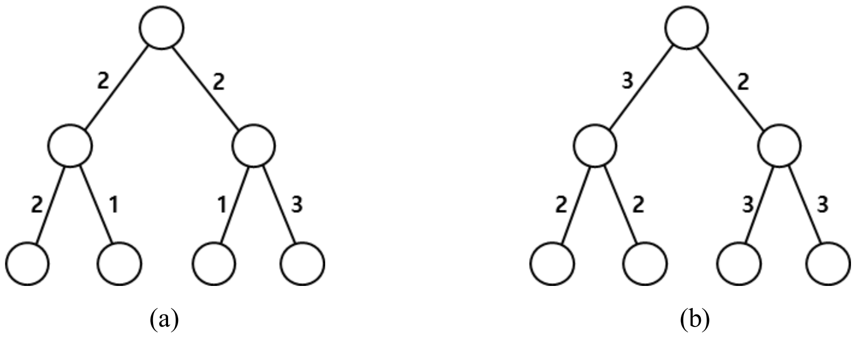


Figure 1. Illustration of increasing edge weights.

Given the weights of all edges in a full binary tree, you are to write a program that increases the weights of certain edges so that all root-to-leaf paths have the same distance and the total edge weights is as small as possible.

Input

Your program is to read from standard input. The input begins with a line containing a positive integer k $(1 \leq k \leq 20)$ which represents the height of a full binary tree. In the second line, all the edge weights are given. The edge weights are given by level-by-level, and left-to-right order in a level. The weight of any edge is between 1 and 1,000, inclusive.

Output

Your program is to write to standard output. Print exactly one line which contains the total edge weights of the tree obtained after increasing edge weights. Note that the weight of a certain edge can be increased to a value larger than 1,000 in some cases.

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

Sample Input 1

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

Output for the Sample Input 1

```
15
```

Sample Input 2

```
1
1 1000
```

Output for the Sample Input 2

```
2000
```

Sample Input 3

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

Output for the Sample Input 3

```
27
```

Sample Input 4

```
2
1 1000 1 1 1000 1000
```

Output for the Sample Input 4

```
5001
```

The 41st Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem A
이진 트리
Time Limit: 1 Second

각 에지에 양수인 가중치가 부여된 높이 k인 포화이진트리가 주어져 있다. 높이 k인 포화이진트리는 2^k개의 리프를 포함하여 $(2^{k+1} - 1)$개의 노드를 가진다. 루트에서 어떤 리프까지의 거리는 루트에서 그 리프까지의 경로상에 있는 모든 에지들의 가중치를 더한 값이다. 이 문제에서는, 어떤 에지들의 가중치를 증가시켜서 루트에서 모든 리프까지의 거리가 같도록 하고, 또한 에지 가중치들의 총합을 최소화 하려고 한다.

예를 들어, 그림 1(a)에 있는 높이 2 인 포화이진트리를 살펴보자. 에지 옆에 있는 수는 그 에지의 가중치를 나타낸다. 이 경우에 대한 답이 그림 1(b)에 나타나 있다. 즉, 루트에서 모든 리프까지의 거리가 5 이고, 에지 가중치들의 총합은 이 경우에 가능한 최소값인 15 이다.

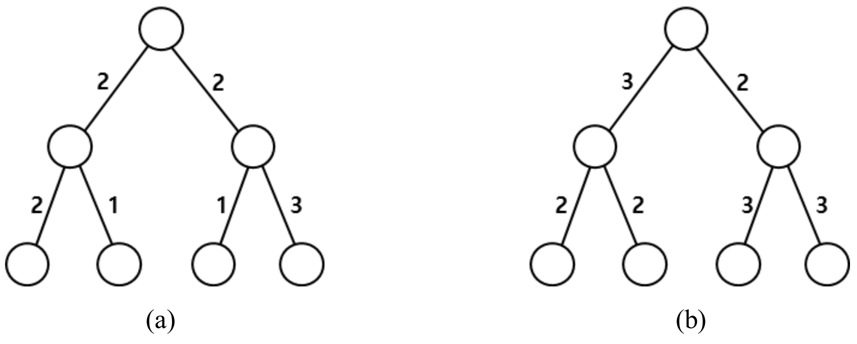


그림 1. 에지 가중치를 증가시키는 예.

포화이진트리에 들어 있는 모든 에지들의 가중치가 주어졌을 때, 어떤 에지들의 가중치를 증가시켜서 루트에서 모든 리프까지의 거리가 같게 하면서 에지 가중치들의 총합이 최소가 되도록 하는 프로그램을 작성하시오.

입력(Input)
입력 데이터는 표준입력을 사용한다. 입력의 첫째 줄에는 포화이진트리의 높이를 나타내는 양의 정수 k $(1 \leq k \leq 20)$가 주어진다. 두 번째 줄에는 모든 에지들의 가중치가 주어진다. 에지들의 가중치는 루트에서 가까운 레벨에 있는 것부터, 같은 레벨에 있는 경우는 왼쪽부터 오른쪽의 순서로 주어진다. 각 에지의 가중치는 1 이상 1,000 이하인 정수로 주어진다.

출력(Output)
출력은 표준출력을 사용한다. 에지들의 가중치를 증가시킨 다음에 얻어지는 트리에 있는 모든 에지들의 가중치들의 총합을 한 줄에 출력한다. 어떤 에지의 가중치는 경우에 따라 1,000 이상의 값으로 증가될 수도 있다는 점에 유의하시오.

다음은 네 개의 테스트 데이터에 대한 입력과 출력의 예이다.

입력 예제 1(Sample Input 1)	**출력 예제 1(Output for the Sample Input 1)**
2 2 2 2 1 1 3	15

입력 예제 2(Sample Input 2)	**출력 예제 2(Output for the Sample Input 2)**
1 1 1000	2000

입력 예제 3(Sample Input 3)	**출력 예제 3(Output for the Sample Input 3)**
3 1 2 1 3 2 4 1 1 1 1 1 1 1 1	27

입력 예제 4(Sample Input 4)	**출력 예제 4(Output for the Sample Input 4)**
2 1 1000 1 1 1000 1000	5001

The 41st Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem B
Diameter
Time Limit: 1 Second

A new city has many buildings. For an efficiency of administration, the city wants to split the buildings into two groups, red and blue ones. A *diameter* of a group is defined as the maximum distance of two buildings in the group, which is an important criterion to measure the geometric extent of the group. The smaller the diameter is, the easier the administration is made. The final goal of this work is to determine a partition such that the sum of the diameters of the two groups is minimized.

More precisely, a building is mapped to a point in the plane. The distance between two points is the Euclidean distance between them. Note that the diameter of a point set is the maximum distance of two points in the set. Given a set P of n distinct points in the plane, you partition P into two subsets P_1 and P_2 such that $P_1 \neq \emptyset$, $P_2 \neq \emptyset$, $P_1 \cup P_2 = P$, $P_1 \cap P_2 = \emptyset$, and the sum of the diameters of P_1 and P_2 is minimized. If a subset consists of only one point, then its diameter is zero. You write a program to compute the minimum diameter sum for P.

For example, nine points with integer coordinates are given in the plane as in Figure 1(a). There are many red-blue partitions. If you partition the points as in Figure 1(b), then the diameter of the blue points is $\sqrt{4^2+3^2} = 5$, and the diameter of the red points is $\sqrt{5^2+1^2} = \sqrt{26}$, thus their sum is $5+\sqrt{26}$. The partition in Figure 1(c) has the sum of the diameters of $4+\sqrt{34}$, which is a bit smaller than the sum in Figure 1(b).

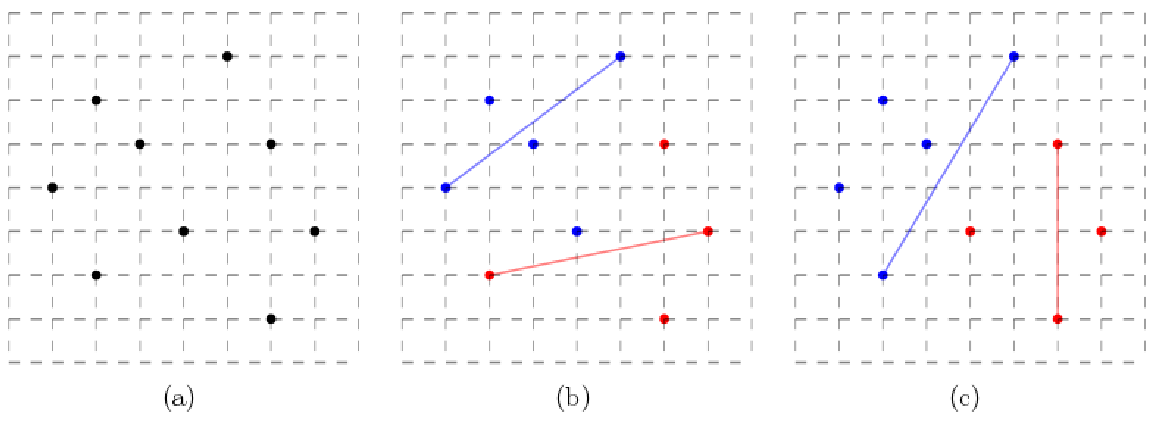


Figure 1. (a) Input points. (b), (c) Two red-blue partitions where the point pair determining the diameter of each group is marked with a line segment with same color of the group.

Input
Your program is to read from standard input. The input starts with a line containing an integer, n ($2 \leq n \leq 5{,}000$), where n is the number of points. In the following n lines, each of the n points in P is given line by line. Each point is represented by two numbers separated by a single space, which are the x-coordinate and the y-coordinate of the point, respectively. The coordinate is an integer between 0 and 10,000, inclusively.

Output

Your program is to write to standard output. Print exactly one line for the input. The line should contain the minimum diameter sum. The output is judged as a correct answer if it is within an absolute error of 10^{-3}.

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

Sample Input 1	Output for the Sample Input 1

3
0 1
3 2
3 0
	2.0000

Sample Input 2	Output for the Sample Input 2

9
2 2
6 1
7 3
6 5
3 5
1 4
2 6
5 7
4 3
	6.0827

Problem C
Frogs
Time Limit: 1 Second

In a hot and dry summer, hungry frogs are headed for "the land of flies" filled with water and foods. They need to cross a river. Normally frogs really swim well, but they are so hungry and tired that they can only walk and jump. There are logs floating on the river and the frogs can a) walk on the logs, or b) jump from one log to another.

As frogs are really hungry and tired, they want to spend as little energy as possible. Walking is free, which means that they spend negligible energy in walking. Jumping is not free: if a frog jumps for a distance x, it spends x^2 units of energy. To reach the land of flies, the total energy spent should be minimized.

Consider the following example. The river and its sides can be represented by a 8×9 grid. Currently, frogs are on the lower side and the land of flies is on the upper side. Note that $(a, b) - (c, d)$ denotes a segment whose endpoints are (a, b) and (c, d). The border of the lower side can be represented by seven segments: $(0,0) - (0,1)$, $(0,1) - (2,1)$, $(2,1) - (2,2)$, $(2,2) - (3,2)$, $(3,2) - (3,1)$, $(3,1) - (7,1)$, and $(7,1) - (7,0)$. The border of the upper side can be represented by eleven segments: $(0,8) - (0,7)$,$(0,7) - (1,7)$, $(1,7) - (1,6)$, $(1,6) - (2,6)$, $(2,6) - (2,7)$, $(2,7) - (4,7)$, $(4,7) - (4,6)$, $(4,6) - (5,6)$, $(5,6) - (5,7)$, $(5,7) - (7,7)$, and $(7,7) - (7,8)$. There are four logs on the river: $(0,3) - (2,3)$, $(6,2) - (4,2)$, $(3,5) - (6,5)$, and $(7,4) - (7,6)$. Assume that all the segments (both for borders and for logs) are parallel either with x-axis or with y-axis. No pair of logs has a common point or a common segment. No log has a common point or a common segment with any border. Again, once a frog lands at one segment, it can move to any point in it freely. Once it arrives at the land of flies, its journey is over.

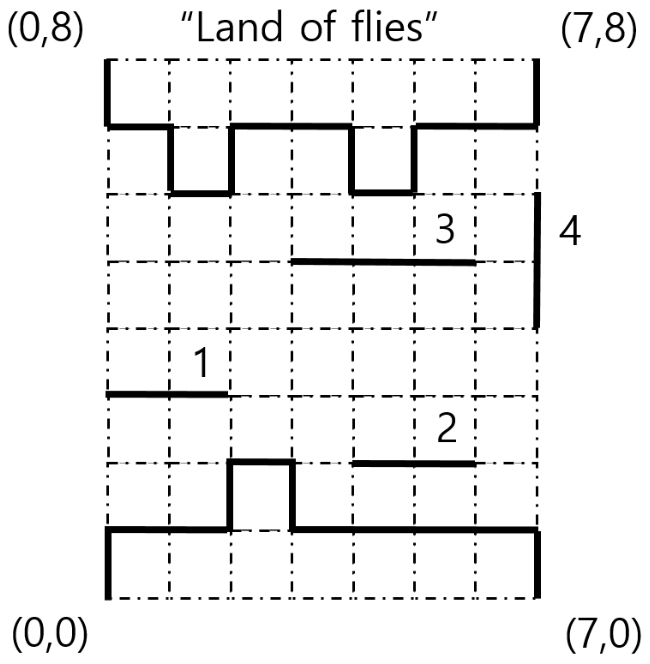


Figure 1. Four logs on the river.

In the example, assume that a frog can jump at most $\sqrt{5}$. One possible path to the land of flies is that a frog first jumps to Log 1, then to Log 3, and to the land of flies. The total energy spent is $1^2 + \sqrt{5}^2 + 1^2 = 7$ and it is easy to show that it is optimal. If a frog can jump at most 2, then it is evident that it cannot reach the land of flies.

You write a program that computes the smallest amount of energy a frog can spend to cross the river.

Input

Your program is to read from standard input. The input consists of several lines. The first line contains two integers n and m which denote the size of grid, $n \times m$ $(3 \leq n, m \leq 5{,}000)$. The next line contains four integers $u, v, w,$ and l $(2 \leq u, v, w \leq 2 * \max(n, m), 1 \leq l \leq \min((n-1)^2, (m-1)^2))$: there are u endpoints on the lower side, and there are v endpoints on the upper side, and there are w logs on the river. Also, a frog can jump at most $\sqrt{l}$. Each of the following u lines contains two integers x and y $(0 \leq x < n,$ $0 \leq y < m)$ representing an endpoint (x, y) of the lower side. These endpoints are given in clockwise order and the bottom left endpoint comes first. Each of the following v lines contains two integers x and y $(0 \leq x < n,$ $0 \leq y < m)$ representing an endpoint (x, y) of the upper side. These endpoints are given in counterclockwise order and the top left endpoint comes first. Each segment linking two neighboring endpoints is parallel either with x-axis or with y-axis. Each of the last w lines contains four integers $x_1, y_1, x_2,$ and y_2 $(0 \leq x_1, x_2 < n,$ $0 \leq y_1, y_2 < m)$ representing a log which is a segment $(x_1, y_1) - (x_2, y_2)$. It is guaranteed that either $x_1 = x_2$ or $y_1 = y_2$. It is also guaranteed that there is no intersection between the lower side and the upper side.

Output

Your program is to write to standard output. Print exactly one line. The line should contain an integer representing the smallest amount of energy required to cross the river. If it is impossible to cross the river, print `-1`.

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

Sample Input 1

```
8 9
8 12 4 5
0 0
0 1
2 1
2 2
3 2
3 1
7 1
7 0
0 8
0 7
1 7
1 6
2 6
2 7
4 7
4 6
5 6
5 7
7 7
7 8
```

Output for the Sample Input 1

```
7
```

0 3 2 3
4 2 6 2
3 5 6 5
7 4 7 6	

Sample Input 2	Output for the Sample Input 2

8 9
8 12 4 4
0 0
0 1
2 1
2 2
3 2
3 1
7 1
7 0
0 8
0 7
1 7
1 6
2 6
2 7
4 7
4 6
6 6
6 7
7 7
7 8
0 3 2 3
4 2 6 2
3 5 6 5
7 4 7 6
	-1

Problem C
개구리

시간 제한: 1 초

어느 덥고 건조한 여름, 배고픈 개구리들이 물과 양식이 풍부한 “파리 나라(the land of flies)”를 향해 여행하고 있다. 여행하는 중 강을 건널 일이 생겼다. 보통의 경우 개구리들은 수영을 잘하지만, 너무 배고프고 피곤한 나머지 개구리는 걷기와 점프만 가능하다. 강 위에 통나무들이 떠다니고 있고, 개구리는 a) 통나무 위에서 걸어 다니거나, b) 한 통나무에서 다른 통나무로 점프할 수 있다.
개구리들은 너무 피곤하고 배고파서, 가능한 한 에너지를 적게 쓰고 싶다. 걸어 다닐 때는 거의 에너지를 쓰지 않기 때문에 걷는 거리는 신경 쓰지 않아도 된다. 반면, 개구리가 거리 x만큼 점프하면, x^2 단위의 에너지를 쓰게 된다. 파리 나라에 도착하는 과정에서 사용된 에너지는 최소화되어야 한다.

다음 예제를 생각해보자. 강과 양쪽 강가는 8×9 크기의 격자로 표현할 수 있다. 처음 개구리는 아래쪽 강가에 있고, 파리 나라는 위쪽 강가에 있다. $(a,b)-(c,d)$는 양 끝점이 (a,b)와 (c,d)인 선분을 의미한다. 아래쪽 강가는 다음과 같은 7 개의 선분으로 표현할 수 있다. $(0,0)-(0,1)$,$(0,1)-(2,1)$, $(2,1)-(2,2)$, $(2,2)-(3,2)$, $(3,2)-(3,1)$, $(3,1)-(7,1)$, $(7,1)-(7,0)$. 위쪽 강가는 다음과 같은 11 개의 선분으로 표현할 수 있다. $(0,8)-(0,7)$,$(0,7)-(1,7)$, $(1,7)-(1,6)$, $(1,6)-(2,6)$, $(2,6)-(2,7)$, $(2,7)-(4,7)$, $(4,7)-(4,6)$, $(4,6)-(5,6)$, $(5,6)-(5,7)$, $(5,7)-(7,7)$, $(7,7)-(7,8)$. 강 위에는 4 개의 통나무가 있다. $(0,3)-(2,3)$, $(6,2)-(4,2)$, $(3,5)-(6,5)$, $(7,4)-(7,6)$. 모든 선분은 (강가이든, 통나무이든) x-축 또는 y-축에 평행하다고 가정하자. 또한 어떤 한 쌍의 통나무도 공통인 영역을 갖지 않고, 통나무와 강가도 공통인 영역을 갖지 않는다. 다시 한번 강조하지만, 개구리는 같은 선분 위라면 어느 위치로든지 에너지를 사용하지 않고 움직일 수 있다. 일단 파리 나라에 개구리가 도착하면 여행은 끝난다.

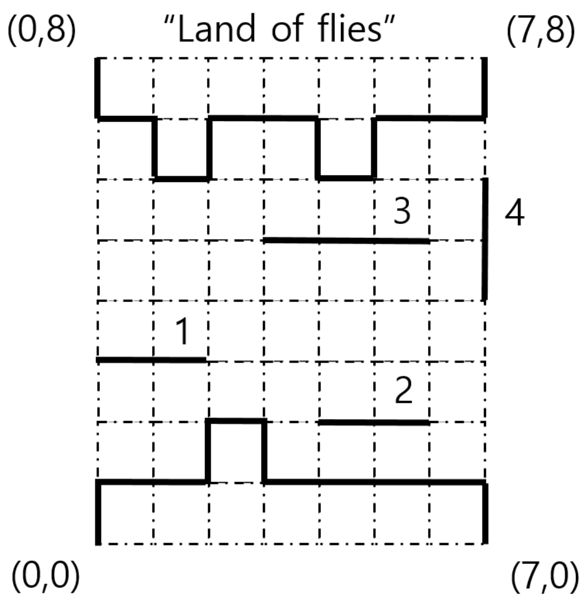


그림 1. 강 위의 네 통나무.

위 예에서, 개구리가 최대로 점프할 수 있는 거리가 $\sqrt{5}$라고 하자. 파리 나라로 도달하는 한가지 경로는 먼저 통나무 1로 점프한 다음, 통나무 3으로 점프하고, 마지막으로 파리 나라로 점프하는 것이다. 이 때 사용된 에너지의 총합은 $1^2 + \sqrt{5}^2 + 1^2 = 7$이며, 이것이 최적이라는 것을 어렵지 않게 보일 수 있다. 만약 개구리가 최대로 점프할 수 있는 거리가 2라면, 파리 나라에 도달할 수 없음을 쉽게 보일 수 있다.

개구리가 강을 건너기 위해 필요한 최소 에너지를 계산하는 프로그램을 작성하시오.

입력

여러분의 프로그램은 표준 입력에서 입력을 받아야 한다. 입력은 여러 줄로 이루어져 있다. 첫 줄에는 두 정수 n과 m이 주어지는데, 이는 격자의 크기 $n \times m$ $(3 \le n, m \le 5{,}000)$을 나타낸다. 다음 줄에는 네 정수 u, v, w, l $(2 \le u, v, w \le 2 * \max(n, m)$, $1 \le l \le \min((n-1)^2, (m-1)^2))$ 이 주어진다. 아래쪽 강가에는 u개의 꼭지점이 있고, 위쪽 강가에는 v개의 꼭지점이 있으며, 강에는 w개의 통나무가 떠 있다. 개구리가 최대로 점프할 수 있는 거리는 $\sqrt{l}$이다. 다음에 오는 u 줄 각각에는 두 정수 x와 y $(0 \le x < n,\ 0 \le y < m)$가 주어지는데, 이는 아래쪽 강가의 한 꼭지점 (x, y)를 나타낸다. 이 꼭지점들은 시계방향 순서대로 주어지며, 가장 아래이면서 가장 왼쪽에 오는 꼭지점이 맨 처음 주어진다. 다음에 오는 v 줄 각각에는 두 정수 x와 y $(0 \le x < n,\ 0 \le y < m)$가 주어지는데, 이는 위쪽 강가의 한 꼭지점 (x, y)를 나타낸다. 이 꼭지점들은 반시계방향 순서대로 주어지며, 가장 위이면서 가장 왼쪽에 오는 꼭지점이 맨 처음 주어진다. 이웃하는 두 꼭지점을 이은 선분은 반드시 x-축에 평행이거나 y-축에 평행이다. 마지막 w 줄 각각에는 네 정수 x_1, y_1, x_2, y_2 $(0 \le x_1, x_2 < n,\ 0 \le y_1, y_2 < m)$가 주어지는데, 이는 선분 $(x_1, y_1) - (x_2, y_2)$로 표현되는 통나무 하나를 표현한다. 반드시 $x_1 = x_2$이거나 $y_1 = y_2$이다. 아래쪽 강가와 위쪽 강가가 교차하지 않는다는 것은 보장된다.

출력

여러분의 프로그램은 표준 출력으로 출력해야 한다. 각 입력에 대해서 정확히 한 줄을 출력한다. 이 줄에는 개구리가 강을 건너가는데 필요한 에너지의 최소값을 출력한다. 만약 개구리가 강을 건너갈 수 없다면, -1 을 출력한다.

다음은 두 테스트 케이스에 대한 입출력 예이다.

예제 입력 1	**예제 입력 1 에 대한 출력**

8 9
8 12 4 5
0 0
0 1
2 1
2 2
3 2
3 1
7 1
7 0
0 8
0 7
1 7
1 6
2 6
2 7
4 7
4 6
5 6
	7

5 7
7 7
7 8
0 3 2 3
4 2 6 2
3 5 6 5
7 4 7 6	

예제 입력 2	예제 입력 2 에 대한 출력

8 9
8 12 4 4
0 0
0 1
2 1
2 2
3 2
3 1
7 1
7 0
0 8
0 7
1 7
1 6
2 6
2 7
4 7
4 6
6 6
6 7
7 7
7 8
0 3 2 3
4 2 6 2
3 5 6 5
7 4 7 6
	-1

Problem D
Message Passing
Time Limit: 1 Second

In International Computer Products Company (ICPC), there is frequently a need to pass an important message from the employer to all the employees. The task of passing a message from ICPC employer to all the employees is accomplished by a series of calls over the telephone along organizational reporting lines. Kim, the employer of ICPC, plans to renovate the massage passing system so that an employee cannot make calls to other employees more than d times. For that purpose, Kim wants to know the number of telephone calls that are made at time t after he started the message passing subject to the constraints that:

(1) Each call involves only two employees.
(2) Each call requires one unit of time.
(3) An employee can participate in only one call per unit of time.
(4) Each employee makes calls to d uninformed employees in consecutive d time units immediately after he or she is informed.

Kim makes a call to an employee only when he starts the message passing. For example, the case when $d = 2$ is shown in the following table.

Time t	Telephone calls	Number of calls at time t
0	Kim starts a message passing by making a call to `A`.	1
1	`A` makes a call to `B` (to pass the message).	1
2	`A` makes a call to `C`. `B` makes a call to `D`.	2
3	`B`, `C`, and `D` make calls to `E`, `F`, and `G`, respectively.	3

In this example, Kim starts the message passing by making a call to `A` at time 0. The call takes one unit of time. At time 1, `A` is the only employee who has the message, and `A` makes a call to an uninformed employee `B`. At time 2, both `A` and `B` have the message; `A` and `B` make calls to uninformed employees `C` and `D`, respectively. Note that each employee cannot make calls to other employees more than two times in this case. As in the above table, three telephone calls are made at time 3.

Given d and t, you are to write a program to compute the number of telephone calls that are made at time t after Kim started the message passing.

Input
Your program is to read from standard input. The input contains two integers, d and t ($2 \leq d \leq 50$, $1 \leq t \leq 2{,}000{,}000{,}000$), where d represents the number of employees to which an employee can make calls, and t represents the time that has been passed since the massage passing started.

Output
Your program is to write to standard output. Print exactly one line as follows: If m is the number of telephone calls that are made at the given time, print m mod 31,991. For example, if m = 32,000, the output should be 9.

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

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

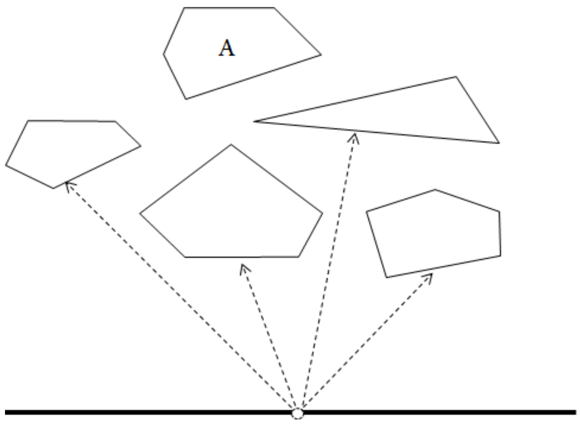
Sample Input 2	Output for the Sample Input 2
`3 3`	`4`

Sample Input 3	Output for the Sample Input 3
`4 3`	`4`

Problem E
Meteor Shower
Time Limit: 1 Second

NSC(Naro Space Center) has just discovered that n large meteorites are falling to Korea. NSC is plannning to blow up the meteorites using laser-guided missile system. In order not to miss a single meteorite, NSC needs to identify the meteorites which are completely blocked by others at a specific moment. We call them *invisible meteorites*.

Each meteorite is represented by a convex polygon. All meteorites are seperated from each other, i.e., any two convex polygons do not intersect each other. The figure below shows an example of a situation with 5 meteorites and a laser-guided missile launcher. In the figure, a meteorite labeled with A is invisible because any point on it can't be touched by a laser beam from the launcher.



Given a list of convex polygons representing meteorites at some moment, write a program to find the number of the meteorites which are invisible from the laser-guided missile launcher.

Input

Your program is to read from standard input. The input starts with a line containing an integer $n (1 \le n \le 100{,}000)$, where n is the number of convex polygons representing meteorites at a specific moment. In the following n lines, each line contains $2m+1$ integers $m, x_1, y_1, x_2, y_2, \ldots, x_m$, and y_m $(3 \le m \le 10^5, -10^8 \le x_i \le 10^8, 1 \le y_i \le 10^8)$, where m is the number of vertices of a convex polygon Q and (x_i, y_i)'s are coordinates of m vertices of Q in the counter-clockwise order. The laser-guided missile launcher is located at $(0,0)$, i.e., the origin of the coordinate system. The total number of vertices of all convex polygons is less than or equal to 10^6. Notice that any two convex polygons do not intersect each other. Also, you may assume that the line connecting any two vertices of all convex polygons does not pass through the origin, i.e., the location of the laser-guided missile launcher.

Output
Your program is to write to standard output. Print exactly one line which contains an integer representing the number of the meteorites which are invisible from the laser-guided missile launcher.

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

Sample Input 1

```
5
3 -2 13 9 12 7 15
5 1 16 -1 18 -5 18 -6 16 -5 14
5 -12 13 -13 11 -11 10 -7 12 -8 13
5 -7 9 -5 7 0 7 1 9 -3 12
5 9 9 6 10 3 9 4 6 9 7
```

Output for the Sample Input 1

```
1
```

Sample Input 2

```
4
4 -800 500 -800 300 700 300 700 500
4 -700 1100 -900 1100 -900 700 -700 700
4 100 700 100 900 -500 900 -500 700
4 300 700 600 700 600 1200 300 1200
```

Output for the Sample Input 2

```
3
```

Problem F
Palindromic
Time Limit: 1 Second

A palindrome is a word which reads the same backward or forward. For example, abba, akasaka, and glenelg are palindromes. Also we can define θ-*palindrome* as follows: given a real number θ $(0 < \theta \leq 1)$, a string w is a θ-palindrome if it can be decomposed as concatenation of three strings, that is, $w = uvu^R$ where u^R is the reversal of u and $\theta \leq 2\frac{|u|}{|w|}$. For example, if $\theta = 0.8$ and $w = ababa$, then w is a θ-palindrome as $u = ab$, $v = a$, and $\theta \leq 2\frac{|u|}{|w|}$. Note that v may be an empty string whose length is zero but u cannot be an empty string.

A string may be represented as concatenation of θ-palindromes. For example, assume that $\theta = 0.5$ and $w = abbaaba$. It is a θ-palindrome itself as $u = ab$, $v = baa$ and $\theta \leq 2\frac{|u|}{|w|} = \frac{4}{7}$. When $\theta = 0.6$, it can be written as concatenation of $abba$ and aba. It is evident that both are θ-palindromes.

Given a string w and a real number θ, you write a program which computes the minimal number of θ-palindromes such that their concatenation is w.

Input
Your program is to read from standard input. The input consists of two lines. The first line contains three integers, n, k, and l $(1 \leq n \leq 10{,}000, 1 \leq k \leq l \leq 100)$ where n is the length of the string w and $\theta = \frac{k}{l}$. The next line contains the string w in English lowercase.

Output
Your program is to write to standard output. Print an integer representing the minimal number of θ-palindromes such that their concatenation is w. If such θ-palindromes do not exist, print `0`.

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

Sample Input 1	Output for the Sample Input 1
7 1 2 abbaaba	1

Sample Input 2	Output for the Sample Input 2
7 3 5 abbaaba	2

Sample Input 3	Output for the Sample Input 3
7 4 5 abcdefg	0

The 41st Annual ACM

International Collegiate Programming Contest

Asia Regional – Daejeon

Nationwide Internet Competition

Problem G

Planar Drawing

Time Limit: 1 Second

In this problem, we consider only finite simple undirected graphs that are connected. A *planar graph* is a graph that can be drawn in the plane in such a way that no two edges cross each other, i.e., its edges intersect only at their end-vertices. Such a drawing is called a *planar drawing* of the graph. Refer to Figure 1 for an example of a planar graph and its planar drawings. It has been known that every planar graph admits a planar drawing such that all edges are straight line segments which do not intersect.

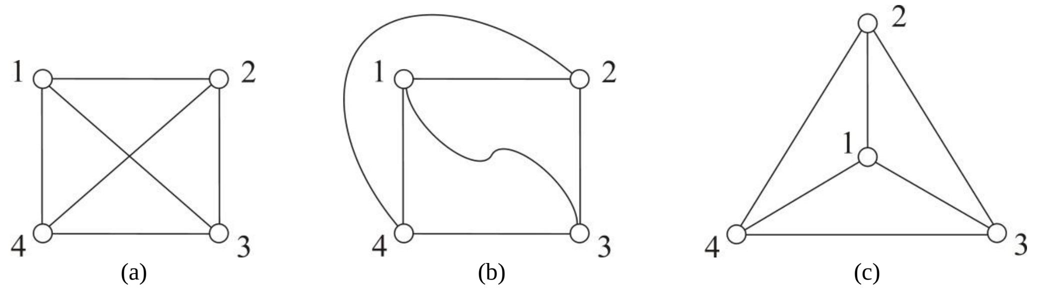


Figure 1. A planar graph and its planar drawings: (a) a complete graph with four vertices; (b) a planar drawing with curved edges; (c) a planar straight-line drawing.

The problem of testing if a given graph is planar is well studied. So, many algorithms for the problem have been designed and implemented, where most of the algorithms run in time linear to the number of vertices in the graph. Sometimes, the users of a planarity testing program expect something more than just a yes or no answer, so as to be sure of the correctness of the output. Recall that the programmers have made mistakes in implementing algorithms that were proven correct. This motivates the study of so-called certifying algorithms.

When the user gives X as an input and the program outputs Y, the user usually has no way of knowing whether Y is a correct output on input X or it has been compromised by a bug. A *certifying algorithm* is an algorithm that produces, with each output, a *certificate* Z that the particular output has not been compromised by a bug. By inspecting the certificate, either manually or by use of a program, the user can convince himself/herself that the output is correct, or reject the output as buggy. The process of checking Z can be automated with a *checker*, which is an algorithm for verifying that Z proves that Y is a correct output for X.

Let us think of the certificates that should be produced by a certifying algorithm for the planarity testing problem. If the input graph is planar, a planar drawing of the graph will be an obvious certificate. If the graph is non-planar, we can utilize Kuratowski's theorem, which states that a graph is planar if and only if it does not contain a subgraph that is a subdivision of K_5 or of $K_{3,3}$. Here, K_5 shown in Figure 2(a) is a complete graph with five vertices; $K_{3,3}$ shown in Figure 2(b) is a complete bipartite graph with three vertices in each bipartition set; a subdivision of a graph is what is obtained by repeatedly subdividing edges by inserting vertices of degree two on them, as illustrated in Figures 2(c) and 2(d). That is, a subdivision of a graph can be obtained if we replace the edges of the graph with independent paths between their end-vertices (so that none of these paths has an inner vertex on another path or in the graph). Thus, if a graph is non-planar, its connected subgraph that is a subdivision of K_5 or of $K_{3,3}$ will be a good certificate.

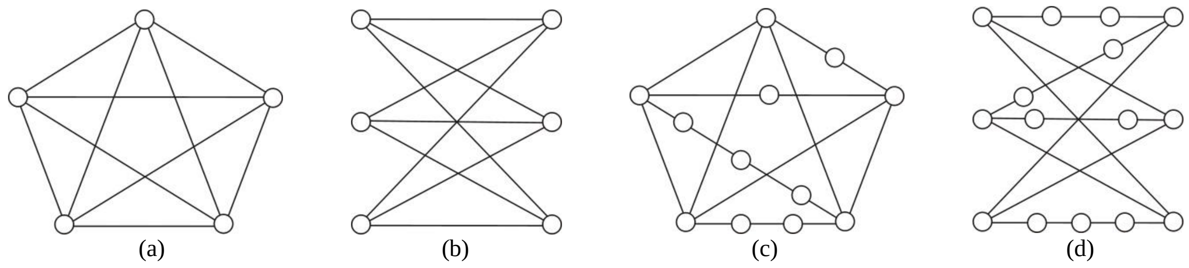
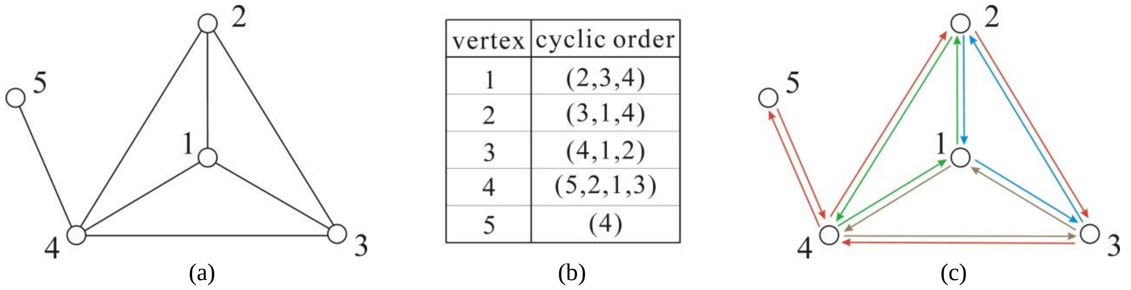


Figure 2. K_5, $K_{3,3}$, and their subdivisions: (a) K_5; (b) $K_{3,3}$; (c) a subdivision of K_5; (d) a subdivision of $K_{3,3}$.

In contrast to the problem of checking if a graph is a subdivision of K_5 or of $K_{3,3}$, it is no easy task to check effectively if a drawing is actually planar. So, a combinatorial embedding defined below is adopted, instead of a planar drawing, as a certificate for the affirmative case. A straight-line drawing of a graph (in which no positions of three vertices are collinear) uniquely defines, for each vertex v of the graph, the cyclic order of the vertices adjacent to v; for convenience, we use clockwise order. The set of all these cyclic orders is called a *combinatorial embedding*. Figure 3 shows a planar drawing and its corresponding combinatorial embedding. In the cyclic order $(2,3,4)$ for instance, the vertex next to 2, 3, and 4, respectively, are 3, 4, and 2.



vertex	cyclic order
1	(2,3,4)
2	(3,1,4)
3	(4,1,2)
4	(5,2,1,3)
5	(4)

Figure 3. A planar drawing and its combinatorial embedding: (a) a planar drawing;
(b) its combinatorial embedding; (c) the four boundary cycles of the combinatorial embedding.

To check whether a combinatorial embedding is indeed planar, we introduce the notion of a boundary cycle in the directed graph obtained by replacing each undirected edge by two directed edges in opposite directions. The *boundary cycle* starting at a directed edge (u, v) is defined as follows: If u' is the vertex next to u in the cyclic order for v, then (v, u') is the next edge of the boundary cycle. We continue in this way until we return to the starting edge (u, v). For example, if we start at a directed edge $(5,4)$ in the directed graph shown in Figure 3(c), then the next edge will be $(4,2)$ because the vertex next to 5 is 2 in the cyclic order $(5,2,1,3)$ for the vertex 4. If we continue, we obtain a boundary cycle: $(5,4) \Rightarrow (4,2) \Rightarrow (2,3) \Rightarrow (3,4) \Rightarrow (4,5) \Rightarrow (5,4)$. It was proven that a combinatorial embedding always leads to a partition of the set of directed edges into a set of boundary cycles, and moreover, for a connected graph with $n > 1$ vertices and m edges, a combinatorial embedding of the graph with f boundary cycles is planar if and only if $n - m + f = 2$. Therefore, it suffices to count the number of boundary cycles and determine if the equation $n - m + f = 2$ holds true.

Suppose we are given an input graph X, as well as the output Y and the certificate Z of a certifying algorithm for the planarity testing problem. Here, Y is of course a yes or no answer; Z is a combinatorial embedding or a subgraph of the input graph X depending on the answer. A checker program may be composed of two modules: (i) one module for testing if the combinatorial embedding Z is planar for the affirmative case, also for testing if the subgraph Z is a subdivision of K_5 or of $K_{3,3}$ for the negative case; (ii) the other module for testing if Z is a combinatorial embedding or a subgraph of the input graph X again depending on the output Y. Your job is to write the first module of the checker program. It is assumed that the graph X has n vertices that are indexed 1 to n.

Input

Your program is to read from standard input. The first line contains an integer indicating the output of a certifying algorithm, where the integer is `1` in the affirmative case while the integer is `-1` in the negative case.

It follows the certificate produced by the certifying algorithm. For the affirmative case, a line containing an integer n, the number of vertices of the graph X, is followed by the combinatorial embedding described through n lines, each contains the cyclic order of vertex k from 1 to n, in the form of d_k, x_1, x_2, …, x_{d_k} representing the cyclic order $(x_1, x_2, \ldots, x_{d_k})$ made of d_k vertices. For the negative case, a line containing the integer n is followed by the connected subgraph produced by the certifying algorithm. The subgraph is described as follows: a line that contains two positive integers n' and m' respectively representing the numbers of vertices and edges of the subgraph is followed by m' lines, each contains two integers u and v that represent an edge between vertex u and vertex v of the subgraph. You may assume that $2 \le n \le 5{,}000$ and also, each vertex of the subgraph is contained in the set $\{1, \ldots, n\}$.

Output

Your program is to write to standard output. Print exactly one line that contains an integer indicating whether the combinatorial embedding Z is planar for the affirmative case, and whether the subgraph Z is a subdivision of K_5 or of $K_{3,3}$ for the negative case. If yes, the integer must be `1`; otherwise `-1`.

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

Sample Input 1

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

Output for the Sample Input 1

```
1
```

Sample Input 2

```
-1
8
6 11
2 3
2 4
2 5
2 6
3 4
3 5
3 6
4 5
4 6
5 7
7 6
```

Output for the Sample Input 2

```
1
```

Sample Input 3

```
-1
10
5 6
5 6
7 6
2 3
7 3
2 5
2 7
```

Output for the Sample Input 3

```
-1
```

The 41st Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem H
Project Team
Time Limit: 1 Second

A team composed of m members is going to carry out a project in n contiguous working days. You are helping the team to set up the vacation schedules of the members. Every member is bound to work for at least p and at most p' working days for the project and, in particular, the number of team members required to work on the i-th working day is between q_i and $q_i{}'$, inclusive, for $i = 1, \dots, n$. Also, each member has a series of vacation plans

$$(d_1, [r_1, r_1']), (d_2, [r_2, r_2']), \dots, (d_k, [r_k, r_k']),$$

which indicate that the member wants to take at least d_i vacation days from the working-day period $[r_i, r_i']$ for each $i = 1, \dots, k$, and wants to work on a day not included in the union of the period $[r_i, r_i']$ over all i, i.e., $\bigcup_{i=1}^{k}\{r: r_i \le r \le r_i'\}$. Note that the vacation days are presumably not fixed in a vacation plan. If a member has a vacation plan $(2, [7{,}9])$ for instance, he/she may take a two-day vacation $\{7{,}8\}$, $\{7{,}9\}$, $\{8{,}9\}$, or happily a three-day vacation $\{7{,}8{,}9\}$; whereas for some vacation plans, say $(2, [3{,}4])$, the vacation days are fixed.

Given the information on the project team and the vacation plans for the team members, you are to write a computer program that determines if it is possible to assign vacation days to each of the members in a way that everyone is happy, subject to the aforementioned constraints. Such an assignment of vacation days is called a vacation schedule. It is assumed that the members are indexed from 1 to m.

For example, suppose you are given $m = 3$, $n = 5$, and $(p, p') = (2, 3)$, as well as $(q_i, q_i{}')$ for working day $i \in \{1, \dots, 5\}$ shown in the table below (left) and the vacation plans for member $j \in \{1{,}2{,}3\}$ shown in the table (center). You can make vacation schedules of the members that satisfy all the constraints, as shown in the table below (right).

i-th day	$(q_i, q_i{}')$
1	$(2, 2)$
2	$(2, 3)$
3	$(1, 2)$
4	$(1, 3)$
5	$(1, 2)$

member j	vacation plans
1	$(2, [1{,}3])$
2	$(2, [2{,}3])$, $(1, [4{,}5])$
3	$(2, [3{,}5])$

member j	vacation days
1	1, 3
2	2, 3, 5
3	4, 5

Input
Your program is to read from standard input. The first line of the input contains four positive integers m, n, p and p' whose meaning has been described above, where $m \le 100$, $n \le 100$, and $p \le p' \le n$. In the following n lines, each line contains two positive integers q_i and $q_i{}'$ for $i = 1$ to n, where $q_i \le q_i{}' \le m$ for all i. Then, m lines follow, where the j-th line contains the vacation plans for the member j. The vacation plans of a member, denoted by $(d_1, [r_1, r_1']), (d_2, [r_2, r_2']), \dots, (d_k, [r_k, r_k'])$, is represented by a sequence of $3k + 1$ positive integers as follows: k, d_1, r_1, r_1', d_2, r_2, r_2', ..., d_k, r_k, r_k'. The number of vacation plans of a member is no more than 20. You may assume that $r_i \le r_i' \le n$ and $d_i \le r_i' - r_i + 1$ for all $i \in \{1, \dots, k\}$, $r_1 < r_2 < \cdots < r_k$, and moreover two periods $[r_a, r_a']$ and $[r_b, r_b']$ are disjoint whenever $a \ne b$, i.e., they do not contain a common working day if $a \ne b$ and $a, b \in \{1, \dots, k\}$.

Output

Your program is to write to standard output. The first line must contain an integer indicating whether or not there exist vacation schedules of the team members that satisfy all the constraints. If yes, the integer must be `1`; otherwise `-1`. When and only when the first line is `1`, it must follow m lines containing, one by one, the vacation schedules for member $j = 1$ to m, where the vacation schedule of a member is described by the number of vacation days followed by the vacation days in ascending order.

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

Sample Input 1

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

Output for the Sample Input 1

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

Sample Input 2

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

Output for the Sample Input 2

```
-1
```

The 41st Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem I
Q-Index
Time Limit: 1 Second

Any Ph.D. candidate in ICPC University should qualify for his/her Ph.D. degree in terms of the number of his/her papers and their citations. For this, the university has defined a quantity, called q-index. To measure the importance of the papers, this index is based on the set of the most cited papers and the number of citations that they have received in other papers. A candidate who has published total $n \geq 1$ papers has q-index k if k of n papers have at least k citations each, and the other $n - k$ papers have at most k citations each.

For example, we suppose that a candidate has published five papers, each cited 8, 4, 5, 3, 10 times by other papers. The candidate has a paper whose citation is at least one, but the other four papers are cited more than once, so the q-index is not 1. For all five papers, there are two papers whose citations are less than 5, so the q-index is not 5. We finally know that the q-index becomes 4 because there are four papers cited at least four times and the other one paper cited at most four times.

Given citation numbers of the papers published by a candidate, you write a program to calculate the q-index.

Input
Your program is to read from standard input. The input starts with a line containing an integer, n $(1 \leq n \leq 1{,}000)$, where n is the number of papers published by a Ph.D. candidate. The next line contains the citation numbers of the n papers, separated by a space. Each citation number is an integer between 0 and $10{,}000$, inclusively.

Output
Your program is to write to standard output. Print exactly one line for the input. The line should contain a non-negative integer representing the q-index.

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

Sample Input 1
```
5
8 4 5 3 10
```
Output for the Sample Input 1
```
4
```

Sample Input 2
```
4
0 0 0 0
```
Output for the Sample Input 2
```
0
```

Sample Input 3
```
6
12 7 6 8 9 10
```
Output for the Sample Input 3
```
6
```

Problem I
Q-인덱스
Time Limit: 1 Second

ICPC 대학의 모든 박사과정 학생은 자신이 발표한 논문과 그 논문들의 인용횟수를 고려한 학위 취득 조건을 만족해야 한다. 이를 위해, ICPC 대학은 q-인덱스라는 값을 정의했다. 이 인덱스는 논문들의 중요도를 측정하기 위해, 가장 많이 인용된 논문들의 개수와 그 논문들의 인용횟수를 이용하여 다음과 같이 정의된다. 한 학생이 발표한 총 $n \geq 1$ 편의 논문 중에서, k번 이상 인용된 논문이 k편이고 나머지 $n-k$ 편의 논문들 인용회수가 각각 k 번 이하라면, 해당 학생의 q-인덱스는 k이다.

예를 들어, 한 학생이 발표한 논문이 총 5편이고, 각 논문의 인용횟수가 8, 4, 5, 3, 10 이라 하자. 한 번 이상 인용된 논문이 1 편 이상이지만 나머지 4편의 논문 중에는 한 번 보다 더 많이 인용된 논문이 존재하기 때문에 q-인덱스는 1 이 아니다. 그리고 모든 논문이 5 번 이상 인용되지 않았기 때문에, 인덱스 값이 5가 될 수도 없다. 이 학생의 q-인덱스는 결국 4가 되는 데, 그 이유는 4번 이상 인용된 논문 4편이 있고, 나머지 1편은 4번 이하의 인용횟수를 갖기 때문이다.

한 학생의 논문들의 인용횟수가 주어지면, 이 학생의 q-인덱스를 계산하는 프로그램을 작성하시오.

입력
프로그램의 입력은 표준 입력으로 받는다. 입력의 첫 줄에는 학생이 발표한 논문의 수 n $(1 \leq n \leq 1{,}000)$이 주어진다. 다음 줄에는 n 개의 논문들에 대한 인용횟수가 빈 칸을 사이에 두고 차례로 주어진다. 각 인용횟수는 0 이상 10,000 이하의 정수 값이다.

출력
표준 출력으로 답을 출력한다. 주어진 입력에 대한 q-인덱스 (음이 아닌 정수) 값 하나를 출력한다.

다음은 세 개의 입력에 대한 출력 값을 나타낸 예제이다.

입력 예제 1	입력 예제 1 에 대한 출력
5 8 4 5 3 10	4

입력 예제 2	입력 예제 2 에 대한 출력
4 0 0 0 0	0

입력 예제 3	입력 예제 3 에 대한 출력
6 12 7 6 8 9 10	6

The 41st Annual ACM

International Collegiate Programming Contest

Asia Regional – Daejeon

Nationwide Internet Competition

Problem J

Railway

Time Limit: 1 Second

There are n persons such that each person commutes from his/her home to his/her office, where his/her home and office are located at different points on a horizontal line. For any two persons A and B, home or office of A can be located at the same point as home or office of B. For the convenience of the persons who commute, we are going to build a railway as a line segment between two points on the horizontal line and operate a rail transport with the stations at points of all homes or offices on the railway. Due to the limited budget, the length of the railway is fixed as d. We want to locate a line segment (railway) L of length d so that the number of persons whose both home and office locations are contained in L is maximized.

Given a positive integer d and n pairs of integers, (h_i, o_i), $1 \le i \le n$, where h_i and o_i are the home location and the office location of person i, respectively, you write a program which prints the maximum number of persons whose both home and office locations are contained in L over all line segments L of length d.

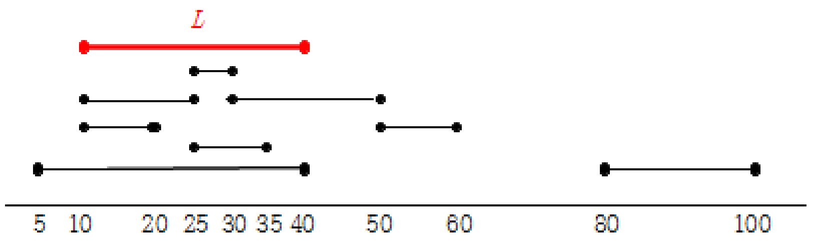


Figure 1. Eight pairs of home and office locations.

Consider an example shown in Figure 1 where $n = 8$, $(h_1, o_1) = (5, 40)$, $(h_2, o_2) = (35, 25)$, $(h_3, o_3) = (10, 20)$, $(h_4, o_4) = (10, 25)$, $(h_5, o_5) = (30, 50)$, $(h_6, o_6) = (50, 60)$, $(h_7, o_7) = (30, 25)$, $(h_8, o_8) = (80, 100)$, and $d = 30$. In this example, the line segment L with red color between 10 and 40 is one of the railway segments which can be located so that the number of persons whose both home and office locations are contained is maximum, thus the answer is 4.

Input

Your program is to read from standard input. The first line contains an integer n ($1 \le n \le 100{,}000$) where n is the number of persons. In each of the following n lines, a pair of integers, (h_i, o_i) are given where h_i and o_i are distinct integers between $-100{,}000{,}000$ and $100{,}000{,}000$, inclusively. The last line has an integer d ($1 \le d \le 200{,}000{,}000$) that represents the length of the railway segment.

Output

Your program is to write to standard output. Print exactly one line for the input. The line should contain an integer representing the maximum number of persons whose both home and office locations are contained in L over all line segments L of length d.

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

Sample Input 1

```
8
5 40
35 25
10 20
10 25
30 50
50 60
30 25
80 100
30
```

Output for the Sample Input 1

```
4
```

Sample Input 2

```
4
20 80
70 30
35 65
40 60
10
```

Output for the Sample Input 2

```
0
```

Sample Input 3

```
5
-5 5
30 40
-5 5
50 40
5 -5
10
```

Output for the Sample Input 3

```
3
```

The 41th Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem J
철로

Time Limit: 1 Second

집과 사무실을 통근하는 n명의 사람들이 있다. 각 사람의 집과 사무실은 수평선 상에 있는 서로 다른 점에 위치하고 있다. 임의의 두 사람 A, B에 대하여, A의 집 혹은 사무실의 위치가 B의 집 혹은 사무실의 위치와 같을 수 있다. 통근을 하는 사람들의 편의를 위하여 일직선 상의 어떤 두 점을 잇는 철로를 건설하여, 기차를 운행하려고 한다. 제한된 예산 때문에, 철로의 길이는 d로 정해져 있다. 집과 사무실의 위치 모두 철로 선분에 포함되는 사람들의 수가 최대가 되도록, 철로 선분을 정하고자 한다.

양의 정수 d와 n 개의 정수쌍, (h_i, o_i), $1 \le i \le n$,이 주어져 있다. 여기서 h_i와 o_i는 사람 i의 집과 사무실의 위치이다. 길이 d의 모든 선분 L에 대하여, 집과 사무실의 위치가 모두 L에 포함되는 사람들의 최대 수를 구하는 프로그램을 작성하시오.

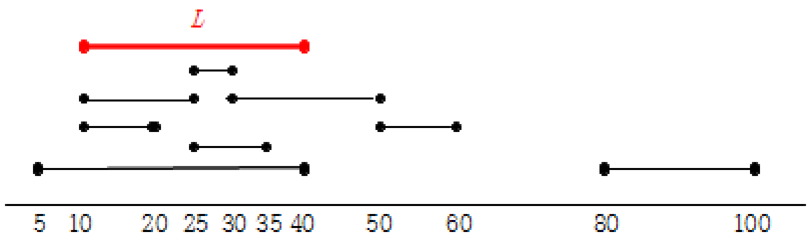


그림 1. 8 명의 집과 사무실의 위치

그림 1 에 있는 예를 고려해보자. 여기서 $n = 8$, $(h_1, o_1) = (5, 40)$, $(h_2, o_2) = (35, 25)$, $(h_3, o_3) = (10, 20)$, $(h_4, o_4) = (10, 25)$, $(h_5, o_5) = (30, 50)$, $(h_6, o_6) = (50, 60)$, $(h_7, o_7) = (30, 25)$, $(h_8, o_8) = (80, 100)$이고, $d = 30$이다. 이 예에서, 위치 10 과 40 사이의 빨간색 선분 L이, 가장 많은 사람들에 대하여 집과 사무실 위치 모두 포함되는 선분 중 하나이다. 따라서 답은 4 이다.

입력(Input)

입력은 표준입력을 사용한다. 첫 번째 줄에 사람 수를 나타내는 양의 정수 n $(1 \le n \le 100{,}000)$이 주어진다. 다음 n개의 각 줄에 정수 쌍 (h_i, o_i)가 주어진다. 여기서 h_i와 o_i는 $-100{,}000{,}000$이상, $100{,}000{,}000$이하의 서로 다른 정수이다. 마지막 줄에, 철로의 길이를 나타내는 정수 d $(1 \le d \le 200{,}000{,}000)$가 주어진다.

출력(Output)

출력은 표준출력을 사용한다. 길이 d의 임의의 선분에 대하여, 집과 사무실 위치가 모두 그 선분에 포함되는 사람들의 최대 수를 한 줄에 출력한다.

다음은 세 개의 입 출력 예제이다.

입력 예제 1

```
8
5 40
35 25
10 20
10 25
30 50
50 60
30 25
80 100
30
```

입력 예제 1 에 대한 출력

```
4
```

입력 예제 2

```
4
20 80
70 30
35 65
40 60
10
```

입력 예제 2 에 대한 출력

```
0
```

입력 예제 3

```
5
-5 5
30 40
-5 5
50 40
5 -5
10
```

입력 예제 3 에 대한 출력

```
3
```

The 41st Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem K
Registration
Time Limit: 1 Second

Print out your ICPC team number and team name.

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 team number, and the second line should contain your full team name, even if your team name contains one or more blanks (a character of ASCII 32).

The following shows sample input and output, where the team number is `123` and the team name is `Your_ICPC_Team_Name`. Notice that no input is given.

Sample Input	Output for the Sample Input
	`123`
	`Your_ICPC_Team_Name`

The 41st Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem K
등록

Time Limit: 1 Second

자신의 ICPC 팀 번호와 팀 이름(team name)을 그대로 출력하는 프로그램을 작성하시오.

Input

이 문제는 입력이 없다.

Output

표준출력(standard output)으로 출력해야 한다. 첫 줄에 자신의 팀 번호, 둘째 줄에 팀 이름을 출력한다. 출력할 팀 이름은 공백문자(ASCII 코드 32 번인 문자)를 포함하더라도, 공백문자를 포함하여 완전한 이름을 출력해야 한다.

다음은 팀 번호가 `123` 번, 팀 이름(team name)이 `Your_ICPC_Team_Name` 인 경우의 입출력 예제이다. 참고로 입력이 없는 것에 주의한다.

Sample Input	Output for the Sample Input
	123 Your_ICPC_Team_Name

Problem L
Trucks
Time Limit: 1 Second

There are n cargo trucks in a line to cross a one lane wide bridge over a river. The order of trucks cannot be changed and the weight of each truck may not be the same. Only w trucks can be on the bridge at the same time. We assume that the length of the bridge is w unit distance and each truck moves a unit distance in a unit time. The sum of the weights of the trucks on the bridge should be less than or equal to L due to the maximum safe load of the bridge. Note that the weights of trucks that aren't fully present on the bridge are not considered in the load calculation of the trucks on the bridge.

For example, let w be 2 and L be 10 and the sequence of the weights of the trucks be [7, 4, 5, 6] and the trucks are crossing from right to left. Then, the earliest time for all the trucks to cross the bridge is 8, as illustrated in Figure 1.

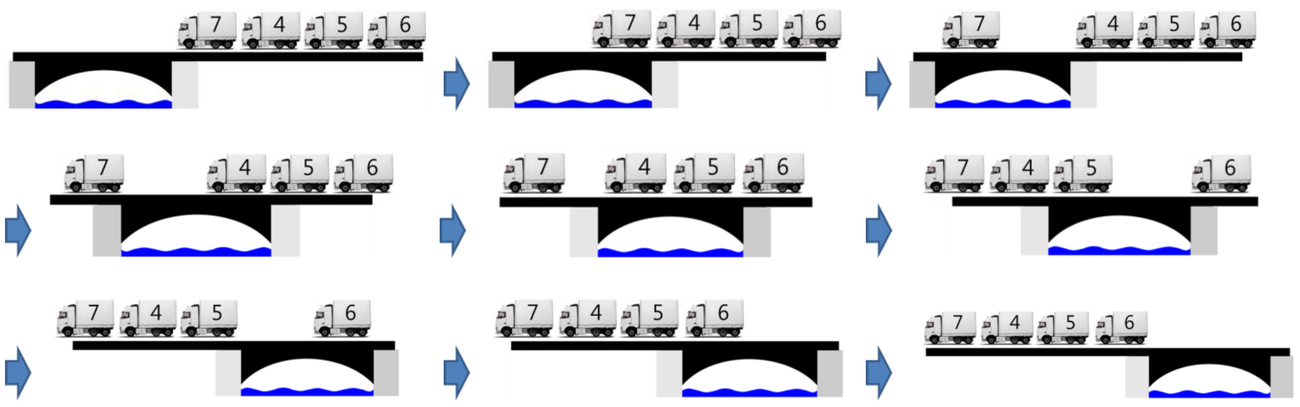


Figure 1. Trucks crossing the bridge.

You are given the maximum affordable number of the trucks on the bridge and the maximum safe load of the bridge and the sequence of the weights of the trucks. Write a program for finding the earliest time for the trucks to cross the bridge.

Input
Your program is to read from standard input. The input consists of two lines. The first line contains three integers n ($1 \leq n \leq 1{,}000$), w ($1 \leq w \leq 100$) and L ($10 \leq L \leq 1{,}000$) where n is the number of trucks to cross the bridge, w is the maximum affordable number of trucks on the bridge and L is the maximum safe load of the bridge. The second line contains n integers $a_1, a_2, \cdots, a_n$ ($1 \leq a_i \leq 10$), where a_i represents the weight of i-th truck.

Output
Your program is to write to standard output. Print the earliest time to cross the bridge.

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

Sample Input 1	Output for the Sample Input 1
4 2 10 7 4 5 6	8

Sample Input 2	Output for the Sample Input 2
1 100 100 10	101

Sample Input 3	Output for the Sample Input 3
10 100 100 10 10 10 10 10 10 10 10 10 10	110

The 41st Annual ACM
International Collegiate Programming Contest
Asia Regional – Daejeon
Nationwide Internet Competition

Problem L
트럭

Time Limit: 1 Second

강을 가로지르는 하나의 차선으로 된 다리가 하나 있다. 이 다리를 n 개의 트럭이 건너가려고 한다. 트럭의 순서는 바꿀 수 없으며, 트럭의 무게는 서로 같지 않을 수 있다. 다리 위에는 단지 w 대의 트럭만 동시에 올라갈 수 있다. 다리의 길이는 w 단위길이(unit distance)이며, 각 트럭들은 하나의 단위시간(unit time)에 하나의 단위길이만큼만 이동할 수 있다고 가정한다. 동시에 다리 위에 올라가 있는 트럭들의 무게의 합은 다리의 최대하중인 L보다 작거나 같아야 한다. 참고로, 다리 위에 완전히 올라가지 못한 트럭의 무게는 다리 위의 트럭들의 무게의 합을 계산할 때 포함하지 않는다고 가정한다.

예를 들어, 다리의 길이 w는 2, 다리의 최대하중 L은 10, 다리를 건너려는 트럭이 트럭의 무게가 [7, 4, 5, 6]인 순서대로 다리를 오른쪽에서 왼쪽으로 건넌다고 하자. 이 경우 모든 트럭이 다리를 건너는 최단시간은 아래의 그림에서 보는 것과 같이 8 이다.

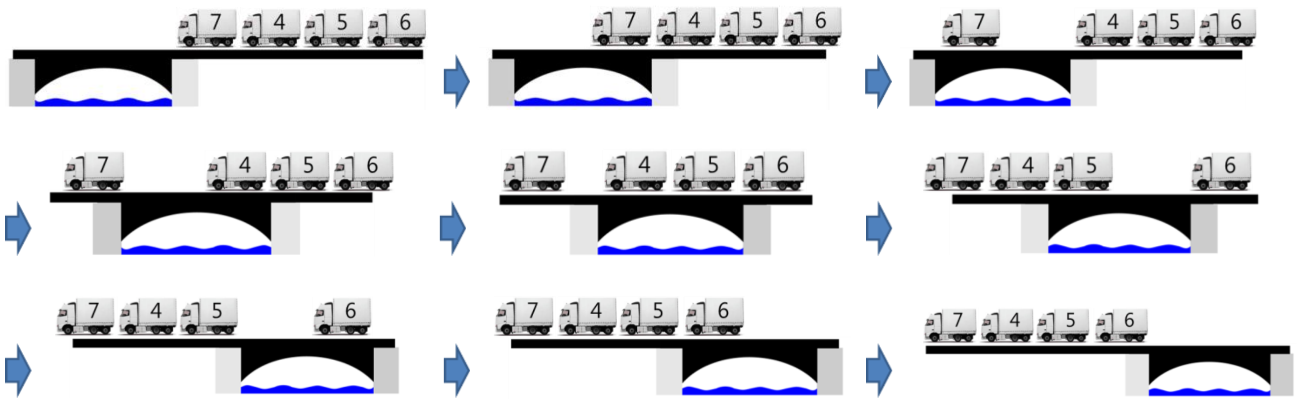


Figure 1. 본문의 예에 대해 트럭들이 다리를 건너는 과정.

다리의 길이와 다리의 최대하중, 그리고 다리를 건너려는 트럭들의 무게가 순서대로 주어졌을 때, 모든 트럭이 다리를 건너는 최단시간을 구하는 프로그램을 작성하라.

입력(Input)

입력 데이터는 표준입력을 사용한다. 입력은 두 줄로 이루어진다. 입력의 첫 번째 줄에는 세 개의 정수 n $(1 \leq n \leq 1{,}000)$, w $(1 \leq w \leq 100)$ and L $(10 \leq L \leq 1{,}000)$이 주어지는데, n은 다리를

건너는 트럭의 수, w는 다리의 길이, 그리고 L은 다리의 최대하중을 나타낸다. 입력의 두 번째 줄에는 n개의 정수 $a_1, a_2, \cdots, a_n$ $(1 \leq a_i \leq 10)$가 주어지는데, a_i는 i번째 트럭의 무게를 나타낸다.

출력(Output)

출력은 표준출력을 사용한다. 모든 트럭들이 다리를 건너는 최단시간을 출력하라.

다음은 두 개의 테스트 데이터에 대한 입력과 출력의 예이다.

입력 예제 1 (Sample Input 1)	**출력 예제 1 (Output for the Sample Input 1)**
4 2 10 7 4 5 6	8

입력 예제 2 (Sample Input 2)	**출력 예제 2 (Output for the Sample Input 2)**
1 100 100 10	101

입력 예제 3 (Sample Input 3)	**출력 예제 3 (Output for the Sample Input 3)**
10 100 100 10 10 10 10 10 10 10 10 10 10	110