ACM International Collegiate Programming Contest 2014

Latin American Regional Contests

November 7th-8th, 2014

Contest Session

This problem set contains 11 problems; pages are numbered from 1 to 14.

This problem set is used in simultaneous contests hosted in the following countries:

Argentina, Bolivia, Brasil, Chile, Colombia, Cuba,
México, Panamá, Perú, República Dominicana and Venezuela

v1.0

General information

Unless otherwise stated, the following conditions hold for all problems.

Program name

1. Your solution must be called *codename*`.c`, *codename*`.cpp` or *codename*`.java`, where *codename* is the capital letter which identifies the problem.

Input

1. The input must be read from standard input.
2. The input consists of a single test case, which is described using a number of lines that depends on the problem. No extra data appear in the input.
3. When a line of data contains several values, they are separated by *single* spaces. No other spaces appear in the input. There are no empty lines.
4. The English alphabet is used. There are no letters with tildes, accents, diareses or other diacritical marks (ñ, Ã, é, Ì, ô, Ü, ç, etcetera).
5. Every line, including the last one, has the usual end-of-line mark.

Output

1. The output must be written to standard output.
2. The result of the test case must appear in the output using a number of lines that depends on the problem. No extra data must appear in the output.
3. When a line of results contains several values, they must be separated by *single* spaces. No other spaces should appear in the output. There should be no empty lines.
4. The English alphabet must be used. There should be no letters with tildes, accents, diareses or other diacritical marks (ñ, Ã, é, Ì, ô, Ü, ç, etcetera).
5. Every line, including the last one, must have the usual end-of-line mark.
6. To output real numbers, round them to the closest rational with the required number of digits after the decimal point. Test case is such that there are no ties when rounding as specified.

Development team

The following persons helped to develop the problem set by creating and improving statements, solutions, test cases and input and output checkers:

Alejandro Strejilevich de Loma, Universidad de Buenos Aires

Bruno Junqueira Adami, Universidade de São Paulo

Cristhian Bonilha, Universidade Tecnológica Federal do Paraná

Fidel I. Schaposnik Massolo, Universidad Nacional de La Plata

Guilherme Albuquerque Pinto, Universidade Federal de Juiz de Fora

Jorge Enrique Moreira Broche, Universidad Central "Marta Abreu" de Las Villas

Pablo Ariel Heiber, Universidad de Buenos Aires

Yonny Mondelo Hernández, Universidad de las Ciencias Informáticas

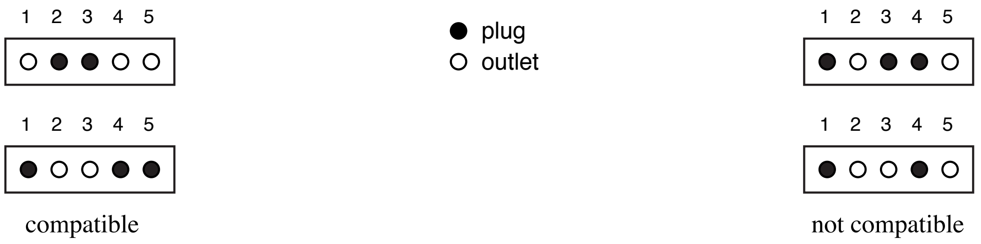
Problem A – Automated Checking Machine

Author: Ricardo Anido, Universidade Estadual de Campinas

The Internet Computer Parts Company (ICPC) is an on-line shop that sells computer parts. Pairs of in-line electrical connectors are among the most popular parts that ICPC sells. However, they are also one of the parts that are returned more often by unsatisfied customers, because due to errors in packaging the connectors sent to the costumers may not be *compatible*.

An in-line connector is composed of five connection points, labelled from 1 to 5. Each connection point of a connector can be either a plug or an outlet. We say two connectors are *compatible* if, for every label, one connection point is a plug and the other connection point is an outlet (in other words, two connectors are compatible if, for every connection point with the same label, a plug and an outlet meet when the two connectors are connected).

The figure below shows examples of two connectors that are compatible and two connectors that are not compatible.



ICPC is introducing a state-of-the-art Automated Checking Machine (ACM), with an optical checker, which will verify whether the two connectors packaged for a customer are indeed compatible. The complex and expensive hardware of the ACM is ready, but they need your help to finish the software.

Given the descriptions of a pair of in-line connectors, your task is to determine if the connectors are compatible.

Input

The first line contains five integers X_i ($0 \leq X_i \leq 1$ for $i = 1, 2, \ldots, 5$), representing the connection points of the first connector in the pair. The second line contains five integers Y_i ($0 \leq Y_i \leq 1$ for $i = 1, 2, \ldots, 5$), representing the connection points of the second connector. In the input, a 0 represents an outlet an a 1 represents a plug.

Output

Output a line with a character representing whether the connectors are compatible or not. If they are compatible write the uppercase letter "Y"; otherwise write the uppercase letter "N".

Sample input 1	**Sample output 1**
1 1 0 1 0 0 0 1 0 1	Y

Sample input 2	**Sample output 2**
1 0 0 1 0 1 0 1 1 0	N

Problem B – Black and white stones

Author: Fidel I. Schaposnik Massolo, Universidad Nacional de La Plata

Shagga and Dolf like to play a game with stones, each of which is either black or white. At the beginning of the game, Dolf arranges all the stones in a single line from left to right. Then, Shagga's goal is to reorder the stones so that all the black stones are to the left of all the white stones. To do this, he can choose any pair of stones of different color and swap their positions, paying A coins to Dolf in the process. However, if the two stones whose positions he is swapping are adjacent, Dolf must give him a refund of B coins, meaning that the operation will effectively cost Shagga only $A - B$ coins.

Shagga is not very bright, so he hasn't realized yet that he will only loose coins while playing this game. However, he is aware of his limitations, so he knows that if he played optimally he would loose fewer coins than he is loosing right now, with his strategy of randomly choosing the stones he swaps in each movement. Therefore, he wants to know the minimum number of coins he will have to pay Dolf in order to get to the desired arrangement of stones, and is threatening to feed you to the goats if you don't help him.

Input

The first line contains two integers A and B ($0 \leq B < A \leq 10^6$), representing respectively the cost of swapping two stones and the value of the refund when swapping adjacent stones. The second line contains a non-empty string S of at most 5000 characters. The i-th character of S indicates the color of the i-th stone, from left to right, in the initial arrangement of the stones. The character is either the uppercase letter "`B`" or the uppercase letter "`W`", indicating respectively a black or a white stone.

Output

Output a line with an integer representing the minimum number of coins Shagga will have to pay Dolf in order to arrange the stones such that all the black ones are to the left of all the white ones.

Sample input 1	**Sample output 1**

`2 1`
`BWWB`
	`2`

Sample input 2	**Sample output 2**

`5 3`
`WBWWBWBWBWBBBWWBBB`
	`27`

Sample input 3	**Sample output 3**

`1000000 0`
`W`
	`0`

Problem C – Counting substhreengs

Author: Pablo Ariel Heiber, Universidad de Buenos Aires

Substrings are strings formed by choosing a subset of contiguous characters from a string. This is well known. A little more obscure is the definition of substhreengs. A substhreeng is a substring which complies to the following additional requirements:

1. It is non-empty, and composed entirely of base 10 digits.
2. Interpreted in base 10 (allowing extra leading zeros), the resulting integer is a multiple of 3.

For instance, the string "`130a303`" contains 9 substhreengs: the substhreeng "`3`" three times, the substhreengs "`30`" and "`0`" twice each, and the substhreengs "`303`" and "`03`" once each. The substring "`30a3`" is not a substhreeng because it is not composed entirely of base 10 digits, while the substring "`13`" is not a substhreeng because 13 is not a multiple of 3.

Notice that two substhreengs are considered different if they are different in length or start at a different position, even if the selected characters are the same.

Given a string, you are asked to count the number of substhreengs it contains.

Input

The input consists of a single line that contains a non-empty string S of at most 10^6 characters. Each character of S is either a digit or a lowercase letter.

Output

Output a line with an integer representing the number of substhreengs contained in S.

Sample input 1	**Sample output 1**
`130a303`	`9`

Sample input 2	**Sample output 2**
`0000000000`	`55`

Sample input 3	**Sample output 3**
`icpc2014regional`	`2`

Problem D – Dividing the names

Author: Leopoldo Taravilse, Universidad de Buenos Aires

The Queen of Nlogonia has decided to move the capital of the queendom to a brand new city called Sortonia. The design for the city is an $N \times N$ grid consisting of N avenues running in the North-South direction and N streets running in the East-West direction. Thus, each avenue intersects every street, and no two streets or two avenues intersect each other.

As the city is almost finished, it is now time to assign names to its streets and avenues. The people of Nlogonia have already voted on the $2 \times N$ names that they want to use, but it hasn't been decided yet which of those will be used for the streets and which for the avenues. The issue is not so simple, because in each crossing there should be a sign identifying the street and the avenue that intersect there, and the Queen has expressly ordered that the letters in these signs ought to be written in gold encrusted with rubies.

Being the official Accountant who Counts the Money (ACM), it is your task to find a way to minimize the total number of letters written in the crossings' signs, for obvious reasons. Luckily, you have thought of a very clever way to achieve this, which is to use abbreviations for the names of the streets and avenues in the signs. The abbreviation for the name of an avenue (respectively a street) is the shortest prefix of its name which is not a prefix of the name of any other avenue (respectively street). Of course, the abbreviation to be used for each name depends on how the set of $2 \times N$ names is split in two disjoint sets of N names to be used for the streets and avenues.

For example, consider the case with $N = 2$ where the four chosen names are "`GAUSS`", "`GALOIS`", "`ERDOS`" and "`EULER`". If the streets are assigned the names "`GAUSS`" and "`GALOIS`", whereas the avenues are assigned the names "`ERDOS`" and "`EULER`", then the abbreviations would be "`GAU`" for "`GAUSS`", "`GAL`" for "`GALOIS`", "`ER`" for "`ERDOS`" and "`EU`" for "`EULER`". With this splitting, the total number of letters to be written in the signs would be 20, as the four intersections would be labeled by "`GAU|ER`", "`GAU|EU`", "`GAL|ER`" and "`GAL|EU`".

However, in the example above it would be more convenient to assign the streets the names "`GAUSS`" and "`ERDOS`", leaving "`GALOIS`" and "`EULER`" for the avenues. Then, the abbreviations would be "`G`" for "`GAUSS`", "`E`" for "`ERDOS`", "`G`" for "`GALOIS`" and "`E`" for "`EULER`", and the total number of letters to be written in the signs would be just 8 (as the intersections would be labeled by "`G|G`", "`G|E`", "`E|G`" and "`E|E`").

Fortunately, the set of names that has been chosen is such that no name in it is a prefix of some other name in the set, thereby ensuring that the scheme you propose will always be feasible. Can you calculate the minimum number of letters to be written in the signs if you split the names optimally?

Input

The first line contains an integer N ($2 \leq N \leq 100$) representing both the number of streets and the number of avenues in Sortonia. Each of the next $2 \times N$ lines contains a non-empty string of at most 18 uppercase letters, indicating one of the names that have been chosen. You may assume that none of the given strings is a prefix of another string in the input.

Output

Output a line with an integer representing the minimum total number of letters to be written in the signs, when the splitting of the names of streets and avenues is chosen optimally.

Sample input 1	**Sample output 1**
2 GAUSS GALOIS ERDOS EULER	8

Sample input 2	**Sample output 2**

`4`
`AA`
`AB`
`AC`
`AD`
`BA`
`BB`
`BC`
`BD`
	`56`

Problem E – Even distribution

Author: Bruno Junqueira Adami, Universidade de São Paulo

Endre has lots of nieces and nephews. Once a year, he takes some of them on a trip to an archipelago where a boat company operates two-way services between some pairs of islands. Since Endre and the children can fly and return directly to or from any of the islands, any trip can be described as a non-empty sequence $i_1, i_2, \ldots, i_n$ of islands, such that each pair of consecutive islands i_j and i_{j+1} have a boat service between them. The first and the last islands of a trip may or may not be the same island, and the islands may be visited more than once during the trip.

Each island in the archipelago produces a different peculiar variety of candy, and greets its visitors by giving each arriving group a fixed number of pieces of candy. Endre does not like candies himself, but the children eat them all almost instantly. To avoid fights, each time the group arrives to an island and receives candies, he evenly distributes them among the children.

You may wonder how Endre always manages to evenly distribute the candies they receive in each island. Well, the answer is actually very simple. Each year, the travel agency sends him the trip plan (the sequence $i_1, i_2, \ldots, i_n$) beforehand. Since he wants to travel with as many of his nieces and nephews as possible, he calculates the maximum number k of kids he can take on the trip without violating the rule about the even distribution of candy. Notice that each trip plan uniquely determines the number of kids to take.

This has been going on for years, and each time Endre ends up taking a different number of kids on the trip. He would like to know how many different numbers of kids he can take on a trip, that is, the number of integers k such that there is a trip plan for which he ends up taking k kids on the trip. Right now Endre is very busy preparing this year's trip. Can you help him with the answer?

Input

The first line contains two integers I and S ($1 \leq I, S \leq 10^4$), representing respectively the number of islands and the number of boat services between them. Islands are identified with distinct integers from 1 to I. The second line contains I integers $C_1, C_2, \ldots, C_I$, where C_i indicates the number of pieces of candy the group receives when arriving at island i ($1 \leq C_i \leq 10^5$ for $i = 1, 2, \ldots, I$). Each of the next S lines describes a different boat service with two integers A and B ($1 \leq A < B \leq I$), representing that it is possible to travel from island A to island B and from island B to island A. No two boat services allow to travel between the same pair of islands.

Output

Output a line with an integer representing the number of integers k such that there is a trip plan for which Endre ends up taking k kids on the trip.

Sample input 1	Sample output 1

2 1
1 9
1 2
	2

Sample input 2	Sample output 2

4 2
1 2 3 4
1 3
1 2
	4

Sample input 3	Sample output 3

4 3
30 42 105 70
2 4
1 2
2 3
	11

Problem F – Fence the vegetables

Author: Pablo Ariel Heiber, Universidad de Buenos Aires

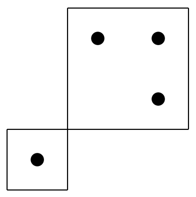
At the early age of 40, Alice and Bob decided to retire. After more than two decades working as examples for networking protocols, game theoretical books and several other texts, they were tired. To remain active, they decided to go into gardening.

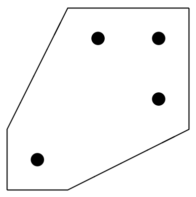
Alice and Bob planted several vegetable plants in a huge field. After finishing, they realized that their plants needed protection from wild animals, so they decided to build a fence around them.

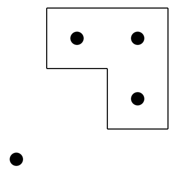
The field is represented as the XY plane, and each vegetable plant as a different point in it. A fence is represented as a polygon in the plane. However, not every polygon is a valid fence. The fence needs to be a single simple polygon with each of its sides parallel to one of the axes. Of course, the polygon must contain all the points representing vegetable plants. A fence too close to the plants or to itself could make it difficult to walk around, so each side of the polygon needs to be at least 1 millimeter away from all plants and all non-adjacent sides.

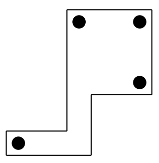
Among all valid fences, Alice and Bob decided to build the one with minimum perimeter, in order to save on fence material. If there are several valid fences with minimum perimeter, they want to build one with minimum area among those, to save time when watering their garden.

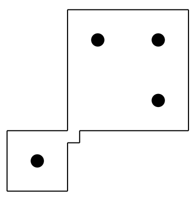
In the following pictures, several different fences are shown in a field with four vegetable plants represented as circles.

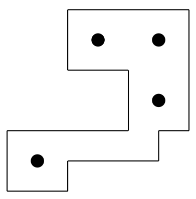

invalid: it is not a simple polygon

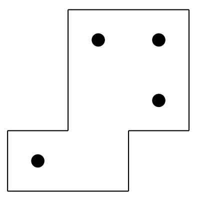

invalid: some sides are not parallel to an axis

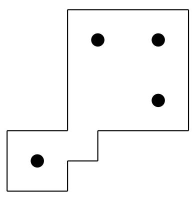

invalid: it does not contain all plants


invalid: it is too close to the plants


invalid: it is too close to itself


suboptimal: it has non-minimum perimeter


suboptimal: it has minimum perimeter but non-minimum area


optimal fence!

Luckily, Alice and Bob's background of participating in rigorous scientific projects made them very thorough with their records: they know the exact location of their plants with millimeter precision. Using this data, help them calculate the perimeter and area of an optimal fence.

Input

The first line contains an integer V ($1 \leq V \leq 10^5$) representing the number of vegetable plants in Alice and Bob's field. Each of the next V lines describes a different vegetable plant with two integers X and Y ($1 \leq X, Y \leq 10^8$), indicating the coordinates of the plant, in millimeters. No two plants have the same location.

Output

Output a line with two integers P and A representing respectively the perimeter in millimeters and the area in squared millimeters of the fence that Alice and Bob want to build.

Sample input 1	**Sample output 1**

`4`
`1 1`
`3 5`
`5 3`
`5 5`
	`24 21`

Sample input 2	**Sample output 2**

`4`
`1 1`
`1 100000000`
`100000000 1`
`100000000 100000000`
	`400000004 10000000200000001`

Sample input 3	**Sample output 3**

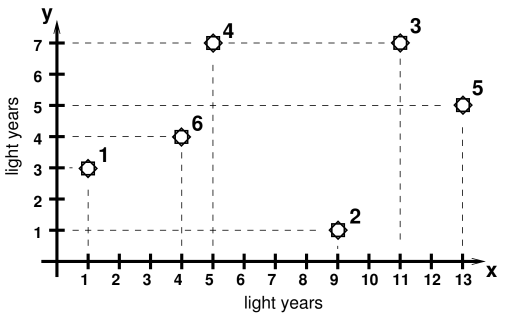
`5`
`50000000 1`
`50000000 99999999`
`1 50000000`
`99999999 50000000`
`50000001 50000001`
	`400000000 399999997`

Problem G – Galaxy collision

Author: Guilherme Albuquerque Pinto, Universidade Federal de Juiz de Fora

The Andromeda galaxy is expected to collide with our Milky Way in about 3.8 billion years. The collision will probably be a merging of the two galaxies, with no two stars actually colliding. That is because the distance between stars in both galaxies is so huge. Professor Andrew is building a computational model to predict the possible outcomes of the collision and needs your help! A set of points in the two dimensional plane is given, representing stars in a certain region of the already merged galaxies. He does not know which stars came originally from which galaxy; but he knows that, for this region, if two stars came from the same galaxy, then the distance between them is greater than 5 light years. Since every star in this region comes either from Andromeda or from the Milky Way, the professor also knows that the given set of points can be separated into two disjoint subsets, one comprising stars from Andromeda and the other one stars from the Milky Way, both subsets with the property that the minimum distance between two points in the subset is greater than 5 light years. He calls this a *good* separation, but the bad news is that there may be many different good separations. However, among all possible good separations there is a minimum number of stars a subset must contain, and this is the number your program has to compute.

For example, the picture illustrates a given set of six points. Professor Andrew cannot tell which stars came from Andromeda, but note that there are four possible good separations: $\{\{1,2,4,5\},\{3,6\}\}$; $\{\{1,2,3,4\},\{5,6\}\}$; $\{\{1,4,5\},\{2,3,6\}\}$; $\{\{1,3,4\},\{2,5,6\}\}$. Therefore, at least two stars must have come from Andromeda, since this is the minimum number of points a subset may have in a good separation.



Input

The first line contains an integer N $(1 \leq N \leq 5 \times 10^4)$ representing the number of points in the set. Each of the next N lines describes a different point with two integers X and Y $(1 \leq X, Y \leq 5 \times 10^5)$, indicating its coordinates, in light years. There are no coincident points, and the set admits at least one good separation.

Output

Output a line with an integer representing the minimum number of points a subset may have in a good separation.

Sample input 1	**Sample output 1**

6
1 3
9 1
11 7
5 7
13 5
4 4
	2

Sample input 2	**Sample output 2**

2
10 10
50 30
	0

Problem H – Help cupid

Author: Pablo Ariel Heiber, Universidad de Buenos Aires

Cupid's job is getting harder, so he is adopting new technologies to help him with his difficult task of matching people into happy couples. He appointed the best programmers in his staff to a new project called Advanced Couples Matching (ACM). For this project, the programmers need to produce an algorithm that takes a set of an even number of N lonely persons and matches them into $N/2$ couples, such that each person is in exactly one couple.

Sadly, the data available about each person is limited. In this modern world, using gender, ethnicity, age or nationality as criteria to form couples is not a sensible option, so the programmers can only use data about the internet connection of each candidate. They decided to focus this stage on time zones. People living in closer time zones are more likely to find time to interact with each other. Thus, the programmers decided to create couples so as to minimize the *total time difference*.

Each time zone is identified by an integer between -11 and 12, inclusive, representing its difference in hours from a particular time zone called Coordinated Universal Time (or UTC). The time difference of two people living in time zones represented by integers i and j is the minimum between $|i - j|$ and $24 - |i - j|$. Given a partition of a set of an even number N of candidates into $N/2$ couples, its total time difference is the sum of the time difference of each couple.

You are asked to write a program that receives as input the time zones of a set of N candidates. The output of the program must be the minimum total time difference among all possible partitions of the set into couples.

Input

The first line contains an even integer N ($2 \leq N \leq 1000$) representing the number of candidates to be coupled. The second line contains N integers $T_1, T_2, \ldots, T_N$ ($-11 \leq T_i \leq 12$ for $i = 1, 2, \ldots, N$) indicating the time zones of the candidates.

Output

Output a line with an integer representing the minimum total time difference among all possible partitions of the set of candidates into couples.

Sample input 1	**Sample output 1**

`6`
`-3 -10 -5 11 4 4`
	`5`

Sample input 2	**Sample output 2**

`2`
`-6 6`
	`12`

Sample input 3	**Sample output 3**

`8`
`0 0 0 0 0 0 0 0`
	`0`

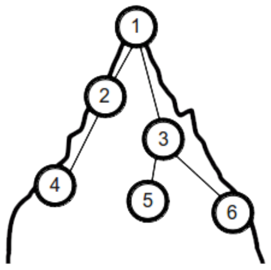
Problem I – Intrepid climber

Author: Cristhian Bonilha, Universidade Tecnológica Federal do Paraná

Who would guess? You climbed the highest mountain of your city. You are so excited about it that you need to tell it to all your friends, and you decided to start with those that are trying to be exactly where you are at this precise moment.

The mountain has N landmarks, and one of them is the top of the mountain, where you are now. Each of your friends that is climbing the mountain is in some other landmark, and you want to visit all of them. There are tracks that connect pairs of landmarks in such a way that there exists exactly one route (that is, a sequence of consecutive tracks) that goes down from the top of the mountain to each other landmark. To visit two friends in two different landmarks, you may have to go down some tracks, climb others, and go down others again. Going down the mountain is "easy", so you do not consume energy when you go down through the tracks. But each time you climb a track, you consume a certain amount of energy. After visiting all your friends, you can just sit and rest.

For example, consider the mountain in the picture below, which has $N = 6$ landmarks. If your friends are in landmarks 5 and 2, you can visit both if you follow the sequence of landmarks $1 \downarrow 2 \uparrow 1 \downarrow 3 \downarrow 5$, where $a \downarrow b$ means that you go down a track from landmark a to landmark b, and $a \uparrow b$ means that you climb a track from landmark a to landmark b. Another possible sequence is $1 \downarrow 3 \downarrow 5 \uparrow 3 \uparrow 1 \downarrow 2$.



Given the tracks between the landmarks, the energy required to climb them, and the landmarks where your friends are, compute the minimum total amount of energy required to visit all your friends from the top of the mountain.

Input

The first line contains two integers N and F ($1 \leq F < N \leq 10^5$), representing respectively the number of landmarks and the number of your friends that are climbing the mountain. Landmarks are identified with distinct integers from 1 to N, being 1 the top of the mountain, where you initially are. Each of the next $N - 1$ lines describes a different track with three integers A, B and C, indicating that there is a track from A to B that goes down and requires an amount C of energy to be climbed ($1 \leq A \leq N$, $2 \leq B \leq N$, $A \neq B$ and $1 \leq C \leq 100$). The next line contains F different integers $L_1, L_2, \ldots, L_F$ ($2 \leq L_i \leq N$ for $i = 1, 2, \ldots, F$) representing the landmarks where your friends are. You may assume that the tracks between landmarks are such that there exists exactly one route that goes down from the top of the mountain to each other landmark.

Output

Output a line with an integer representing the minimum total amount of energy required to visit all your friends starting from the top of the mountain.

Sample input 1	**Sample output 1**

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

Sample input 2	**Sample output 2**

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

Sample input 3	**Sample output 3**

`4 2`
`1 4 1`
`1 3 1`
`4 2 2`
`2 4`
	`0`

Problem J – Journey through the kingdom

Author: Fidel I. Schaposnik Massolo, Universidad Nacional de La Plata

The kingdom of Quadradonia is divided into provinces forming a grid-like pattern of R rows and C columns. Legend has it many wonderful things await discovery in some of the provinces, although it's unclear if you can actually find the elusive solid form of water stories call "ice", or if it's just dragons.

You are planning a trip through the kingdom to find out, but the roads are dangerous so you have to be very careful. To go from one province to another you would like to use the convenient escorted carriage system managed by the Interprovincial Communication & Peregrination Company (ICPC). In each province, the ICPC provides a heavily guarded carriage for you to travel to any other province in a rectangle containing it, at the same flat rate (which may however vary from one province to another). More formally, at the province in the i-th row and the j-th column you can rent an escorted carriage for a cost of V_{ij}, allowing you to safely travel to any province at most R_{ij} rows away from row i, and at most C_{ij} columns away from column j (that is, having row number i' and column number j' with $|i - i'| \le R_{ij}$ and $|j - j'| \le C_{ij}$).

In your journey you want to visit N provinces $p_1, p_2, \ldots, p_N$, in that order. Wandering around looking for adventures is an expensive business and your budget is limited, so you would like to spend as little as possible in transportation. Therefore, you would like to calculate the minimum cost of each leg of your trip, that is, the minimum cost of the carriages you have to rent to go from province p_k to province p_{k+1}, for $k = 1, 2, \ldots, N - 1$.

Input

The first line contains three integers R, C and N, representing respectively the number of rows, the number of columns and the number of provinces you want to visit ($1 \le R, C \le 500$ and $2 \le N \le 5$). Rows are numbered from 1 to R and columns are numbered from 1 to C. The next $3 \times R$ lines describe ICPC's escorted carriage system by means of three groups of R lines each, with each line containing C integers. In the i-th line of the first group, the j-th number represents the cost V_{ij} of renting a carriage in the province in row i and column j, while the corresponding numbers in the second and third group represent respectively R_{ij} and C_{ij} ($1 \le V_{ij} \le 1000$, $0 \le R_{ij} \le R$ and $0 \le C_{ij} \le C$, for $i = 1, 2, \ldots, R$ and $j = 1, 2, \ldots, C$). The next N lines describe the provinces $p_1, p_2, \ldots, p_N$ you want to visit, in the same order you want to visit them. The k-th of these lines describes the province p_k with two integers I_k and J_k, indicating that p_k is in row I_k and column J_k ($1 \le I_k \le R$ and $1 \le J_k \le C$ for $k = 1, 2, \ldots, N$).

Output

Output a line with $N - 1$ integers representing the minimum cost of each leg of your trip, or the value -1 if it is impossible to travel using ICPC's escorted carriage system for that leg. More precisely, for $k = 1, 2, \ldots, N - 1$, the k-th number must be the minimum cost of the carriages you have to rent to go from province p_k to province p_{k+1} using ICPC's escorted carriage system, or the value -1 if it is impossible to travel from province p_k to province p_{k+1} with this system.

Sample input 1	**Sample output 1**

3 4 5
1 2 1 1
1 5 3 4
1 1 6 3
1 2 3 3
3 3 1 2
0 0 0 1
1 4 0 1
2 3 0 1
4 1 3 1
1 1
3 4
1 1
2 2
2 2
	3 -1 1 0

Problem K – Knights of the Round Table

Author: Jorge Enrique Moreira Broche, Universidad Central “Marta Abreu” de Las Villas

Every month King Arthur celebrates a High Council meeting. The K knights attending these meetings are known as The Knights of the Round Table, probably because they sit at a huge round oak table having K seats and a big throne with a sword and a stone carved in its back.

For today's meeting, each knight was given a number between 1 and K that indicates the seat he must take during the meeting. Seats are numbered clockwise from 1 to K, seat numbered 1 being the first to the left of the big throne. Obviously, the king himself was not given a number because he sits on the throne. King Arthur's squire made sure that no two knights got the same number so there should be no problems.

As usual, the king was the first to enter the council room today. According to the protocol rules, he sat on his throne and prepared to receive the K knights that must enter and sit one by one. After the first D knights arrived, the king noted that some of them could have sat on wrong seats, because they were distracted talking about who would win the next tournament. What a mess! King Arthur's squire promptly intervened and gave instructions to the remaining $K - D$ knights. Each of them must enter the council room and try to sit on his rightful seat; if his seat is already occupied, the knight must walk clockwise around the table and sit on the first unoccupied seat he finds. Thus, the final distribution of knights around the table depends on the order in which they enter the room.

King Arthur is now interested in knowing the number of distinct distributions of the K knights around the table given the seats occupied by the first D knights. Two distributions are considered distinct if there is at least one knight who sits on different seats in both distributions.

As the Royal Advisor in Combinatorics and other Mathematics (or Royal ACM) the task is assigned to you. You need to provide an answer within five hours at risk of loosing the king's favor. Hurry up!

Input

The first line contains two integers K ($1 \leq K \leq 10^6$) and D ($1 \leq D \leq 10^5$), representing respectively the number of knights and the number of distracted knights. Each of the next D lines describes a different distracted knight with two integers A and B ($1 \leq A, B \leq K$), indicating that the knight who was assigned seat A actually sat on seat B. It is granted that no two knights sat on the same seat.

Output

Output a line with an integer representing the number of distinct distributions of the K knights around the table. This number can be rather large so output the remainder of dividing it by $10^9 + 7$.

Sample input 1	**Sample output 1**
3 1 1 2	2

Sample input 2	**Sample output 2**

5 4
5 5
1 2
2 3
3 4
	1

Sample input 3	**Sample output 3**

8 3
3 3
4 8
2 4
	2