Harry: "But Hagrid. How am I going to pay for all of this? I haven't any money."

Hagrid: "Well there's your money, Harry! Gringotts, the wizard bank! Ain't no safer place. Not one. Except perhaps Hogwarts."

— Rubeus Hagrid to Harry Potter.

Gringotts Wizarding Bank is the only bank of the wizarding world, and is owned and operated by goblins. It was created by a goblin called Gringott. Its main offices are located in the North Side of Diagon Alley in London, England. In addition to storing money and valuables for wizards and witches, one can go there to exchange Muggle money for wizarding money. The currency exchanged by Muggles is later returned to circulation in the Muggle world by goblins. According to Rubeus Hagrid, other than Hogwarts School of Witchcraft and Wizardry, Gringotts is the safest place in the wizarding world.

The text above is quoted from Harry Potter Wiki. But now Gringotts Wizarding Bank is not safe anymore. The stupid Dudley, Harry Potter's cousin, just robbed the bank. Of course, uncle Vernon, the drill seller, is behind the curtain because he has the most advanced drills in the world. Dudley drove an invisible and soundless drilling machine into the bank, and stole all Harry Potter's wizarding money and Muggle money. Dumbledore couldn't stand with it. He ordered to put some magic lights in the bank rooms to detect Dudley's drilling machine. The bank can be considered as a $N \times M$ grid consisting of $N \times M$ rooms. Each room has a coordinate. The coordinates of the upper-left room is (1,1) , the down-right room is (N, M) and the room below the upper-left room is (2,1).... A 3×4 bank grid is shown below:

(1,1)	(1,2)		(1,4)
(2,1)			
(3,1)			(3,4)

Some rooms are indestructible and some rooms are vulnerable. Dudely's machine can only pass the vulnerable rooms. So lights must be put to light up all vulnerable rooms. There are at most fifteen vulnerable rooms in the bank. You can at most put one light in one room. The light of the lights can penetrate the walls. If you put a light in room (x, y), it lights up three rooms: room (x, y), room $(x-1, y)$ and room $(x, y+1)$. Dumbledore has only one special light whose lighting direction can be turned by 0 degree,90 degrees, 180 degrees or 270 degrees. For example, if the special light is put in room (x, y) and its lighting direction is turned by 90 degrees, it will light up room (x, y), room $(x, y+1)$ and room $(x+1, y)$. Now please help Dumbledore to figure out at least how many lights he has to use to light up all vulnerable rooms.

Please pay attention that you can't light up any indestructible rooms, because the goblins there hate light.

Input

There are several test cases.

In each test case:

The first line are two integers N and M, meaning that the bank is a $N \times M$ grid($0 < N, M \leq 200$).

Then a $N \times M$ matrix follows. Each element is a letter standing for a room. '#' means a indestructible room, and '.' means a vulnerable room.

The input ends with $N = 0$ and $M = 0$.

Output

For each test case, print the minimum number of lights which Dumbledore needs to put.

If there are no vulnerable rooms, print '0'.

If Dumbledore has no way to light up all vulnerable rooms, print '-1'.

Sample Input

```
2 2
##
##
2 3
#..
..#
3 3
###
#.#
###
0 0
```

Sample Output

```
0
2
-1
```

Harry Potter has some precious. For example, his invisible robe, his wand and his owl. When Hogwarts school is in holiday, Harry Potter has to go back to uncle Vernon's home. But he can't bring his precious with him. As you know, uncle Vernon never allows such magic things in his house. So Harry has to deposit his precious in the Gringotts Wizarding Bank which is owned by some goblins. The bank can be considered as a $N \times M$ grid consisting of $N \times M$ rooms. Each room has a coordinate. The coordinates of the upper-left room is (1,1) , the down-right room is (N, M) and the room below the upper-left room is (2,1).... A 3×4 bank grid is shown below:

(1,1)	(1,2)		(1,4)
(2,1)			
(3,1)			(3,4)

Some rooms are indestructible and some rooms are vulnerable. Goblins always care more about their own safety than their customers' properties, so they live in the indestructible rooms and put customers' properties in vulnerable rooms. Harry Potter's precious are also put in some vulnerable rooms. Dudely wants to steal Harry's things this holiday. He gets the most advanced drilling machine from his father, uncle Vernon, and drills into the bank. But he can only pass though the vulnerable rooms. He can't access the indestructible rooms. He starts from a certain vulnerable room, and then moves in four directions: north, east, south and west. Dudely knows where Harry's precious are. He wants to collect all Harry's precious by as less steps as possible. Moving from one room to another adjacent room is called a 'step'. Dudely doesn't want to get out of the bank before he collects all Harry's things. Dudely is stupid.He pay you \$1,000,000 to figure out at least how many steps he must take to get all Harry's precious.

Input

There are several test cases.

In each test cases:

The first line are two integers N and M, meaning that the bank is a $N \times M$ grid($0 < N, M \leq 100$).

Then a $N \times M$ matrix follows. Each element is a letter standing for a room. '`#`' means a indestructible room, '`.`' means a vulnerable room, and the only '`@`' means the vulnerable room from which Dudely starts to move.

The next line is an integer K ($0 < K \leq 4$), indicating there are K Harry Potter's precious in the bank.

In next K lines, each line describes the position of a Harry Potter's precious by two integers X and Y, meaning that there is a precious in room (X, Y).

The input ends with $N = 0$ and $M = 0$.

Output

For each test case, print the minimum number of steps Dudely must take. If Dudely can't get all Harry's things, print '`-1`'.

Sample Input

```
2 3
##@
#.#
1
2 2
4 4
#@##
....
####
....
2
2 1
2 4
0 0
```

Sample Output

```
-1
5
```

In the ancient three kingdom period, Zhuge Liang was the most famous and smart military leader. His enemy was Sima Yi, the military leader of Kingdom Wei. Sima Yi always looked stupid when fighting against Zhuge Liang. But it was Sima Yi who laughed to the end.

Zhuge Liang had led his army across the mountain Qi to attack Kingdom Wei for six times, which all failed. Because of the long journey, the food supply was a big problem. Zhuge Liang invented a kind of bull-like or horse-like robot called "Wooden Bull & Floating Horse"(in abbreviation, WBFH) to carry food for the army. Every WBFH had a password lock. A WBFH would move if and only if the soldier entered the password. Zhuge Liang was always worrying about everything and always did trivial things by himself. Since Ma Su lost Jieting and was killed by him, he didn't trust anyone's IQ any more. He thought the soldiers might forget the password of WBFHs. So he made two password cards for each WBFH. If the soldier operating a WBFH forgot the password or got killed, the password still could be regained by those two password cards.

Once, Sima Yi defeated Zhuge Liang again, and got many WBFHs in the battle field. But he didn't know the passwords. Ma Su's son betrayed Zhuge Liang and came to Sima Yi. He told Sima Yi the way to figure out the password by two cards. He said to Sima Yi:

> "A password card is a square grid consisting of $N \times N$ cells.In each cell,there is a number. Two password cards are of the same size. If you overlap them, you get two numbers in each cell. Those two numbers in a cell may be the same or not the same. You can turn a card by 0 degree, 90 degrees, 180 degrees, or 270 degrees, and then overlap it on another. But flipping is not allowed. The maximum amount of cells which contains two equal numbers after overlapping, is the password. Please note that the two cards must be totally overlapped. You can't only overlap a part of them."

Now you should find a way to figure out the password for each WBFH as quickly as possible.

Input

There are several test cases.

In each test case:

The first line contains a integer N, meaning that the password card is a $N \times N$ grid ($0 < N \leq 30$).

Then a $N \times N$ matrix follows, describing a password card. Each element is an integer in a cell.

Then another $N \times N$ matrix follows, describing another password card.

Those integers are all no less than 0 and less than 300.

The input ends with $N = 0$.

Output

For each test case, print the password.

Sample Input

```
2
1 2
3 4
5 6
7 8
2
10 20
30 13
90 10
13 21
0
```

Sample Output

```
0
2
```

Apollonius of Perga (ca. 262 BC - ca. 190 BC) was a Greek geometer and astronomer. In his noted work Epaphai, he posed and solved such a problem: constructing circles that are tangent to three given circles in a plane. Two tangent circles can be internally or externally tangent to each other, thus Apollonius's problem generically have eight solutions.

Now considering a simplified case of Apollonius's problem: constructing circles that are externally tangent to two given circles, and touches a given point(the given point must be on the circle which you find, can't be inside the circle). In addition, two given circles have no common points, and neither of them are contained by the other, and the given point is also located strictly outside the given circles. You should be thankful that modern mathematics provides you with plenty of useful tools other than euclidean geometry that help you a lot in this problem.

Input

The first line of input contains an integer T ($T \leq 200$), indicating the number of cases.

Each ease has eight positive integers x_1, y_1, r_1, x_2, y_2, r_2, x_3, y_3 in a single line, stating two circles whose centres are (x_1, y_1), (x_2, y_2) and radius are r_1 and r_2 respectively, and a point located at (x_3, y_3). All integers are no larger than one hundred.

Output

For each case, firstly output an integer S, indicating the number of solutions.

Then output S lines, each line contains three float numbers x, y and r, meaning that a circle, whose center is (x, y) and radius is r, is a solution to this case. If there are multiple solutions ($S > 1$), outputing them in any order is OK. Your answer will be accepted if your absolute error for each number is no more than 10^{-4}.

Sample Input

```
1
12 10 1 8 10 1 10 10
```

Sample Output

```
2
10.00000000 8.50000000 1.50000000
10.00000000 11.50000000 1.50000000
```

A random number generator (RNG) is a computational or physical device designed to generate a sequence of numbers or symbols that lack any pattern, i.e. appear random. Many of these have existed since ancient time, including dices, coin flipping, the shuffling of playing cards, and many other techniques. These methods depend on the measurement of some physical phenomenon which is expected to be random, and are still widely used today. On the other hand, deterministic computational algorithms were introduced into random number generation. Despite such algorithms' ability to produce apparently random results, they are in fact determined by a shorter initial value, known as a seed or key. These algorithms are often called pseudorandom number generators. They can also be called RNG customarily, but actually differ with real RNG significantly.

Now considering a simple RNG, whose algorithm has two positive integer parameters A, B and a prime parameter M ($2 \leq A, B < M \leq 10^{18}$). To run the algorithm, a seed X_0 ($0 \leq X_0 < M$) is required, and the algorithm produces a integer sequence X_n satisfying the condition $X_n = (A \times X_{n-1} + B) \bmod M$ for any positive integer n.

An application implemented this algorithm. This application has another two parameters S ($S \leq M$) and K ($K \leq 10^5$), and will use this RNG in such a way. Firstly, the application generates the first K integers in the random sequence including the seed, and these numbers modulo S are stored in another number sequence D, i.e. $D_i = X_i \bmod S$ for any integer i in $[0, K-1]$. Then, another random integer X_K is produced, and the application chooses the $((X_K \bmod K) + 1)$-th number in sequence D, i.e. $D_{X_k \bmod K}$ as its output C. If an output C ($0 \leq C < S$) is observed in a certain run, and parameters A, B, M, S, K is known, your task is determining a possible X_0 which leads to the output C.

Input

There are at most 200 test cases. Each test case is a single line containing six integers, A, B, M, S, K, and C, seperated by space. The meaning of these numbers are described above.

The input is ended by '`0 0 0 0 0 0`'

Output

You should output an integer for each test case, indicating a possible X_0.

Sample Input

```
2 2 97 5 3 2
2 2 97 5 3 3
0 0 0 0 0 0
```

Sample Output

```
2
8
```

Go is a proverbial board game originated in China. It has been proved to be the most difficult board game in the world. “The rules of Go are so elegant, organic, and rigorously logical that if intelligent life forms exist elsewhere in the universe, they almost certainly play Go.” said Emanuel Lasker, a famous chess master.

A Go board consists of 19 horizontal lines and 19 vertical lines. So there are 361 cross points. At the beginning, all cross points are vacant.

Go is played by two players. The basic rules are:

1. One player owns black stones and the other owns white stones.

2. Players place one of his stones on any vacant cross points of the board alternately. The player owns black stones moves first.

3. Vertically and horizontally adjacent stones of the same color form a chain.

4. The number of vacant points adjacent (vertically or horizontally) to a chain is called the liberty of this chain. Once the chain has no liberty, it will be captured and removed from the board.

5. While a player place a new stone such that its chain immediately has no liberty, this chain will be captured at once unless this action will also capture one or more enemy's chains. In that case, the enemy's chains are captured, and this chain is not captured.

In effect, Go also has many advanced and complex rules. However, we only use these basic rules mentioned above in this problem.

Now we are going to deal with another game which is quite similar to Go. We call it “Infinite Go”. The only difference is that the size of the board is no longer 19 times 19 — it becomes infinite. The rows are numbered 1, 2, 3, ..., from top to down, and columns are numbered 1, 2, 3, ..., from left to right. Notice that the board has neither row 0 nor column 0, which means even though the board is infinite, it has boundaries on the top and on the left.

In this problem, we are solving the problem that, given the actions of two players in a set of Infinite Go, find out the number of remaining stones of each player on the final board.

Input

The input begins with a line containing an integer T ($1 \le T \le 20$), the number of test cases.

For each test case, the first line contains a single integer N ($1 \le N \le 10000$), the number of stones placed during this set. Then follows N lines, the i-th line contains two integer X and Y ($1 \le X, Y \le 2,000,000,000$), indicates that the i-th stone was put on row X and column Y (i starts from 1). The stones are given in chronological order, and it is obvious that odd-numbered stones are black and even-numbered ones are white.

Output

For each test case, output two integers Nb and Nw in one line, separated by a single space. Nb is the number of black stones left on the board, while Nw is the number of white stones left on the board.

Sample Input

```
1
7
5 5
4 5
3 5
3 4
4 4
3 3
4 6
```

Sample Output

```
4 2
```

There are some apple trees in a farm. An apple tree can be described as a connected graph which has n nodes and $n-1$ edges. The apples are the nodes and the branches are the edges. Every edge is assigned a value denoting the length of the branch.

Now in the farm come a lot of ants, which are going to enjoy the delicious apples. The ants climb the tree one by one. Every ant would choose a node as the starting node and another node as the ending node, then it would crawl alone the unique path from the starting node to the ending node. The distance between two nodes is defined as the XOR sum of lengths of all the edges in the unique path between them. Every ant wants to crawl along such a path which the distance is as large as possible. But two ants cannot crawl from the same starting node to the same ending node. You should calculate the distance which the k-th ant crawled.

Note that the starting node and the ending node cannot be the same for an ant.

Input

The input consists of several test case.

For each test case, the first line contain an integer n denoting the number of nodes.

The next $n-1$ lines each contains three integers x, y, z, denoting that there exists an edge between node x and node y and its length is z. The nodes are numbered from 1 to n.

The next line contain a integer m denoting the number of queries.

In the next m lines, each line contains an integer k denoting that you need to calculate the distance of the k-th ant.

The input ends with $n = 0$.

Specifications: $1 \le n, m \le 100000$, $1 \le x, y \le n$, $0 \le z \le 10^{18}$, $1 \le k \le 200000$.

Output

For each query, output the answer. If such path does not exist, just output '`-1`'.

Notes:

In the first test case, the first ant may crawl from node 2 to node 3, and the second ant may crawl from node 3 to node 2, and the 5-th ant may crawl from node 1 to node 3.

The distance of the 5-th ant can be calculated by 2 `xor` 3 = 1.

Sample Input

```
3
1 2 2
3 2 3
3
1
2
5
5
1 3 7
2 1 3
4 3 6
5 3 1
3
1
8
1000
0
```

Sample Output

```
3
3
1
7
6
-1
```

Long long ago, there was an ancient rabbit kingdom in the forest. Every rabbit in this kingdom was not cute but totally pugnacious, so the kingdom was in chaos in season and out of season.

n rabbits were numbered form 1 to n. All rabbits' weight is an integer. For some unknown reason, two rabbits would fight each other if and only if their weight is NOT co-prime.

Now the king had arranged the n rabbits in a line ordered by their numbers. The king planned to send some rabbits into prison. He wanted to know that, if he sent all rabbits between the i-th one and the j-th one(including the i-th one and the j-th one) into prison, how many rabbits in the prison would not fight with others.

Please note that a rabbit would not fight with himself.

Input

The input consists of several test cases.

The first line of each test case contains two integer n, m, indicating the number of rabbits and the queries.

The following line contains n integers, and the i-th integer W_i indicates the weight of the i-th rabbit.

Then m lines follow. Each line represents a query. It contains two integers L and R, meaning the king wanted to ask about the situation that if he sent all rabbits from the L-th one to the R-th one into prison ($1 \leq n, m, W_i \leq 200000$, $1 \leq L \leq R \leq n$).

The input ends with $n = 0$ and $m = 0$.

Output

For every query, output one line indicating the answer.

Note: In the second case, the answer of the 4-th query is 2, because only 1 and 5 is co-prime with other numbers in the interval [2,6].

Sample Input

```
3 2
2 1 4
1 2
1 3
6 4
3 6 1 2 5 3
1 3
4 6
4 4
2 6
0 0
```

Sample Output

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

Alice and Bob are playing “Gems Fight!”:

There are Gems of G different colors, packed in B bags. Each bag has several Gems. G different colors are numbered from color 1 to color G.

Alice and Bob take turns to pick one bag and collect all the Gems inside. A bag cannot be picked twice. The Gems collected are stored in a shared cooker.

After a player, we name it as X, put Gems into the cooker, if there are S Gems which are the same color in the cooker, they will be melted into one Magic Stone. This reaction will go on and more than one Magic Stone may be produced, until no S Gems of the same color remained in that cooker. Then X owns those new Magic Stones. When X gets one or more new Magic Stones, he/she will also get a bonus turn. If X gets Magic Stone in a bonus turn, he will get another bonus turn. In short, a player may get multiple bonus turns continuously.

There will be B turns in total. The goal of “Gems Fight!” is to get as more Magic Stones than the opponent as possible.

Now Alice gets the first turn, and she wants to know, if **both of them act the optimal way**, what will be the difference between the number of her Magic Stones and the number of Bob's Magic Stones at the end of the game.

Input

There are several cases(≤ 20).

In each case, there are three integers at the first line: G, B, and S. Their meanings are mentioned above.

Then B lines follow. Each line describes a bag in the following format: $n\ c_1\ c_2 \ldots c_n$

It means that there are n Gems in the bag and their colors are color c_1, color c_2 …and color c_n respectively ($0 \leq B \leq 21$, $0 \leq G \leq 8$, $0 < n \leq 10$, $S < 20$).

There may be extra blank lines between cases. You can get more information from the sample input.

The input ends with $G = 0$, $B = 0$ and $S = 0$.

Output

One line for each case: the amount of Alice's Magic stones minus the amount of Bob's Magic Stones.

Hint: For the first case, in turn 2, bob has to choose at least one bag, so that Alice will make a Magic Stone at the end of turn 3, thus get turn 4 and get all the three Magic Stones.

Sample Input

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

3 2 2
3 2 3 1
3 1 2 3

0 0 0
```

Sample Output

```
3
-3
```

DRD loves playing computer games, especially Tower Defense games. Tower Defense is a famous computer game with a number of variations. In general, you are to build some defense towers to guard your territory in this game.

However, in most Tower Defense games, your defending towers will not attack each other. You will see the shells flying through your towers and finally hit the target on your screen. DRD thinks it to be absurd, and he designed a new tower defense game.

In DRD's game, you have two kinds of defending tower, heavy tower and light tower. You can put the tower on a grid with N rows and M columns and each cell in the grid can hold one tower at most. Both two kinds of towers can attack the cells in the same column or the same row as it is located in, and your towers may attack each other. Moreover, light towers should not be attacked by other towers while heavy towers can be attacked by at most one other tower.

You can put some of your towers (at least one tower) in the grid to build a tower formation satisfying the restriction above. And now, DRD wants you to calculate that how many different tower formations could be designed. Note that all the towers of the same type are considered to be identical. While the answer could be quite large, you should output the number $\mathrm{mod}(10^9 + 7)$.

Input

There are multiple test cases in the input. The first line of the input file is an integer T demonstrating the number of test cases. $(0 < T \leq 200)$.

For each test case, there is only one line containing 4 integers, N, M, P and Q, meaning that the grid has N rows and M columns, and you have P heavy towers and Q light towers. You do not have to put all the towers in the grid $(1 \leq N, M \leq 200, 0 \leq P, Q \leq 200)$.

Output

For each test case, output the number of different formations $\mathrm{mod}(10^9 + 7)$ in a single line.

Sample Input

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

Sample Output

```
4
10
144
```

A new candy factory opens in pku-town. The factory import M machines to produce high quality candies. These machines are numbered from 1 to M.

There are N candies need to be produced. These candies are also numbered from 1 to N. For each candy i, it can be produced in any machine j. It also has a producing time (s_i, t_i), meaning that candy i must start producing at time s_i and will finish at t_i. Otherwise if the start time is $p_i (s_i < p_i < t_i)$ then candy will still finish at t_i but need additional $K \times (p_i - s_i)$ cost. The candy can't be produced if p_i is greater than or equal to t_i. Of course one machine can only produce at most one candy at a time and can't stop once start producing.

On the other hand, at time 0 all the machines are in their initial state and need to be "set up" or changed before starting producing. To set up Machine j from its initial state to the state which is suitable for producing candy i, the time required is C_{ij} and cost is D_{ij}. To change a machine from the state suitable for candy i_1 into the state suitable for candy i_2, time required is $E_{i_1 i_2}$ and cost is $F_{i_1 i_2}$.

As the manager of the factory you have to make a plan to produce all the N candies. While the sum of producing cost should be minimized.

Input

There are multiple test cases.

For each case, the first line contains three integers N $(1 \le N \le 100)$, M $(1 \le M \le 100)$, K $(1 \le K \le 100)$. The meaning is described above.

Then N lines follow, each line contains 2 integers s_i and t_i $(0 \le s_i < t_i < 100000)$.

Then N lines follow, each line contains M integers, the j-th integer of the i-th line indicating C_{ij} $(1 \le C_{ij} \le 100000)$.

Then N lines follow, each line contains M integers, the j-th integer of the i-th line indicating D_{ij} $(1 \le D_{ij} \le 100000)$.

Then N lines follow, each line contains N integers, the i_2-th integer of the i_1-th line indicating $E_{i_1 i_2}$ $(1 \le E_{i_1 i_2} \le 100000)$.

Then N lines follow, each line contains N integers, the i_2-th integer of the i_1-th line indicating $F_{i_1 i_2}$ $(1 \le F_{i_1 i_2} \le 100000)$.

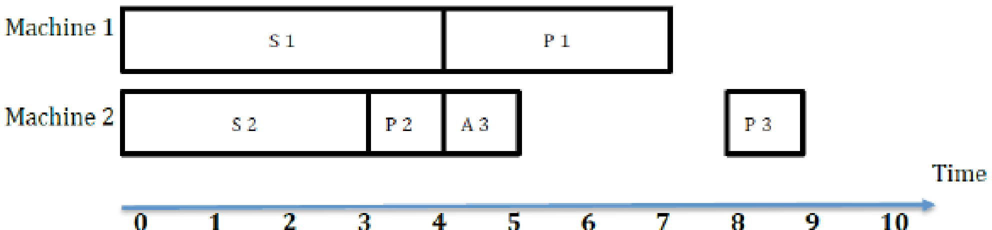
Since the same candy will only be produced once, E_{ii} and F_{ii} are meaningless and will always be '-1'.

The input ends by $N = 0$, $M = 0$, $K = 0$. Cases are separated with a blank line.

Output

For each test case, if all of M candies can be produced, output the sum of minimum producing cost in a single line. Otherwise output '-1'.

Hint: For the first example, the answer can be achieved in the following way:



In the picture, Si represents setting up time for candy i, Ai represents changing time for candy i and Pi represents producing time for candy i.

So the total cost includes:

- setting up machine 1 for candy 1, costs 2
- setting up machine 2 for candy 2, costs 3
- changing state from candy 2 to candy 3, costs 5
- late start of candy 2, costs 1

Sample Input

```
3 2 1
4 7
2 4
8 9
4 4
3 3
3 3
2 8
12 3
14 6
-1 1 1
1 -1 1
1 1 -1
-1 5 5
5 -1 5
5 5 -1

1 1 2
1 5
5
5
-1
-1

0 0 0
```

Sample Output

```
11
-1
```