Problem A
Convert a Boolean Formula
Time limit: 2 seconds

In this problem, we consider two algebraic structures: Boolean algebra and finite field GF(2).

In mathematical logic, Boolean algebra is the algebra in which the values of the variables are the truth values **false** or **true**, usually represented by 0 and 1, respectively. There are 3 operations: $\vee$, $\wedge$, and $\neg$.

$$0 \vee 0 = 0, \quad 0 \vee 1 = 1 \vee 0 = 1 \vee 1 = 1.$$

$$0 \wedge 0 = 0 \wedge 1 = 1 \wedge 0 = 0, \quad 1 \wedge 1 = 1.$$

$$\neg 0 = 1, \quad \neg 1 = 0.$$

Boolean formulas are formulas containing Boolean variables with operators $\vee$, $\wedge$, and $\neg$. For simplicity, in this problem we consider only *simple* Boolean formulas of the form

$$x_1 \vee x_2 \vee \cdots \vee x_n,$$

where each x_i is a variable or the negation of a variable. Each variable is different, and no variable and its negation will both appear in the formula.

Since computer keyboards may not have $\vee$, $\wedge$, and $\neg$ keys, we will replace $\vee$ by "`+`", $\wedge$ by "`x`" and $\neg$ by appending a `'` after the variable.

In finite field GF(2), there are only two elements 0 and 1. The two operations in GF(2) are $+$ and $\times$.

$$0 + 0 = 1 + 1 = 0, \quad 0 + 1 = 1 + 0 = 1.$$

$$0 \times 0 = 0 \times 1 = 1 \times 0 = 0, \quad 1 \times 1 = 1.$$

If you are not familiar with GF(2), you can think of GF(2) as integers. After each addition (or multiplication) of 2 integers, the final result is the remainder of the sum (or the product) divided by 2.

A polynomial P in GF(2) is a polynomial with coefficient 0 and 1. The addition and multiplication of 2 polynomials are the same as ordinary polynomials, except that each final coefficient is the remainder of the coefficient divided by 2. For example,

$$(a + b)^2 = a^2 + 2ab + b^2 = a^2 + b^2 = a + b.$$

Note that $x^2 = x$ in GF(2).

Let B be a Boolean formula and P be a GF(2) polynomial with the same set of variables. B and P are equivalent, if for every assignment of the valuables, B and P always have the same value. The following table gives a useful set of equivalent B's and P's.

B	P
x	x
$x \vee y$	$x + y + xy$
$x \wedge y$	xy
x'	$1 + x$

Based on the above table, write a program to convert a simple Boolean formula into its equivalent polynomial in GF(2).

Input File Format

There are more than one test cases in the input file. Each test case contains a simple Boolean formula of n, $1 \leq n \leq 8$, valuables in $\{a, b, c, d, e, f, g, h\}$. For simplicity, if a Boolean formula contains $n < 8$ variables, the first n variables in the list a, b, c, d, e, f, g, h will be used. For example, if $n = 3$, then the variables are a, b, c. Furthermore, the variables will appear in increasing order. For example, $a + b' + c$, not $b' + a + c$ or $c + a + b'$. There will be no spaces in each input line. The last line of the input file contains a single 0. After scanning this line, your program should stop.

Output Format

For each Boolean formula, print its equivalent polynomial in GF(2). The polynomial should be simplified and expressed as the sum of products. For example, express a polynomial as $a + ab + ac + bc$, not $(a + b)(a + c)$. If the polynomial is a constant, print the constant. Otherwise, follow the following rules in printing out the polynomial.

1. If the coefficient of a term is 0, do not print this term.
2. For each term with coefficient 1, print only the product of the valuables. For example, print `ab`, not `1ab`.
3. The product of the valuables must be printed in alphabetical order. For example, print `abc`, not `cab`.
4. If the polynomial contains more than 1 term:
   (a) Print the terms with fewer number of variables first. For example, print $1 + a + ab$, not $a + ab + 1$.
   (b) For terms with the same number of variables, print them in dictionary order. For example, print $ab + bc$, not $bc + ab$.
   (c) Insert the string " `+` " (a space followed by "`+`" followed by another space) between two adjacent terms.

Sample Input

```
a+b'
a+b+c
a'+b+c
0
```

Output for the Sample Input

```
1 + b + ab
a + b + c + ab + ac + bc + abc
1 + a + ab + ac + abc
```

Problem B
Extended One-Max Problem
Time limit: 2 seconds

An extended one-max problem is a problem for maximizing and minimizing the value of the objective function $f(s)$ of a binary string $s = \langle s_1, s_2, \ldots, s_n \rangle$, where the objective function is defined as

$$f(s) = ((m \textbf{ mod } 10) + 1) \times (((\textbf{B2D}(s) + 1) \textbf{ mod } (m + 1)) + 1)$$

where $m = \sum_{i=1}^{n} s_i$, $s_i \in \{0, 1\}$, $n \leq 20$ and $\textbf{B2D}(s)$ is a function for converting the binary string s to a decimal value.

Input Format

The input is a number n representing the number of bits in the binary string $s = \langle s_1, s_2, \ldots, s_n \rangle$, as the sample input shows.

Output Format

The output contains two binary strings s^a and s^b each of which is followed by its objective value. The difference is in that the first string s^a is the one with the maximum objective value $f(s^a) = \max f(s)$ while the second string s^b is the one with the minimum objective value $f(s^b) = \min f(s)$, as the sample output shows.

Sample Input

```
3
0
```

Output for the Sample Input

```
011: 6
000: 1
```

Problem C
Sister Cities
Time limit: 1 second

The ACM kingdom has n cities, numbered 0, 1, ..., $n-1$, where n is even. No cities cross the equatorial. So each city is either in the northern or southern hemisphere, but not both.

The ACM Queen wants to form sister cities. So she asks each city c to propose a list ℓ_c of exactly 2^k cities, with which c is willing to have sister city relationship. Willingness is known to be mutual: If a city d is in ℓ_c, then c is in ℓ_d. Of course, k is a non-negative integer specified by the ACM Queen. To facilitate communication between the northern and southern hemispheres, ℓ_c has to contain only cities not in the hemisphere containing c, for each city c. I.e., if c is in the northern (southern) hemisphere, then all cities in ℓ_c must be in the southern (northern, respectively) hemisphere.

Please help the ACM Queen order all cities as c_0, c_1, ..., c_{n-1} such that for each even number $0 \le i \le n-2$, c_i and c_{i+1} are willing to be sister cities (i.e., c_i is in $\ell_{c_{i+1}}$). Sensibly, the ordering is intended to make c_i and c_{i+1} sister cities for each even number $0 \le i \le n-2$. Note that c_0, c_1, ..., c_{n-1} should be a permutation of 0, 1, ..., $n-1$.

Should two or more solutions exist, please just output one of them.

Input File Format

The first line is n, where $n \le 100000$. The second line is $n \cdot 2^{k-1}$, where $k \le 16$. For each pair (c, d) of cities such that c and d are willing to be sister cities (i.e., c is in ℓ_d), there is exactly one remaining line giving c and d (or, in the opposite order, d and c). Two numbers in a line are separated by space(s).

Output Format

Order all cities as c_0, c_1, ..., c_{n-1} such that for each even number $0 \le i \le n-2$, c_i and c_{i+1} are willing to be sister cities. Then output c_i and c_{i+1} in one line, for each even number $0 \le i \le n-2$. So there should be $n/2$ lines of output. Note that there may be many correct outputs.

Sample Input

```
10
20
9 0
9 2
9 8
9 3
1 2
1 8
3 1
4 1
8 6
6 3
6 4
0 6
5 3
5 4
5 0
5 2
7 4
0 7
2 7
8 7
```

Output for the Sample Input

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

Problem D
Multiple of 9
Time limit: 1 second

Input an array of $\boldsymbol{N}$ ($1 \leq N \leq 10^5$) *non-negative* integers where each element $A[i]$ is a multiple of **3** and $A[i] \in \{0, 3, 6, 9\}$. Please write a computer program to find **the maximum decimal number, which is a multiple of 9 by concatenating some of the array elements**.

Input File Format

This first line will be the number of the test cases.

In each test case, the first line is the number of elements $\boldsymbol{N}$ and the second line will be the array of elements, where two consecutive elements are separated by a space.

Output Format

Output **the maximal decimal number, which is a multiple of 9 and two consecutive elements are separated by a space**. The output of each test case should be placed in a separate line. If no such solution exists, then output -1.

Sample Input

```
2
5
3 6 3 9 3
2
6 6
```

Output for the Sample Input

```
9 3 3 3
-1
```

Problem E
Spy Network
Time limit: 2 seconds

The international association C.I.A. (abbrev for Computer Intelligence Association) is invited by the intelligent agency SIS to conduct a computer-aided investigation regarding the evergrowing spying activities for Biegnij. The goal of this investigation is to discover the exact structural information of the lurking spy network, which is an undirected graph, so that proper countermeasure can be devised.

It is known that, in order for the spy network to remain hidden, communication of the network is designed to be done in a rather local manner. Not a single spy agent holds the complete information of the network. Instead, each spy agent holds a device that can establish secure communication with the following three types of recipients: (1) the agent himself/herself, (2) the agents he/she is connected to in the network, and (3) the agents his/her neighboring spy agents are connected to in the network. The device owned by each spy agent stores the list of the three types of recipients as described.

During a successful secret operation, the intelligent agents in SIS were able to access the devices of the spy agents and acquire the complete list of recipients of each device from its preinstalled backdoor in the government-designed software. Unfortunately, they were not able to identify to which spy each device belongs. This says, they were not able to identify the owner of each list. The good news is that, the top secret agent of SIS, Bomd, has revealed in the same operation that the underlying structure of the spy network is in fact a tree.

Given the complete list of recipients of each spying device, assuming that the spy agents in the network are indexed from 1 to n, your task in this problem is to help C.I.A. uncover the connections(edges) of the spy network.

Input Format

The input contains multiple testcases.

The first line of each testcase consists of an integer n, where $2 \leq n \leq 1000$, which is the number of spy agents in the network. Then there are n lines, each of which describes the complete list of recipients of one spying device. Each of these n lines starts with an integer, which is the length of the list, followed by the indexes of the recipients in the list.

It is guaranteed that the input lists of each testcase correspond to a valid spy network.

A testcase starting with $n = 0$ indicates the end of input.

Output Format

For each testcase, print the edges of the spy networks, one at a line. Each line should describe an edge and consist of the indexes of its endpoints, separated by a space.

Both the edges and the indexes of the endpoints of the edges can be printed in any order. If there are multiple solutions, you may print any of them.

Sample Input

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

Sample Output for the Sample Input

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

Problem F
Bike Lane Planning
Time limit: 2 seconds

The NCPC Recreational bike-only park has many bike paths that connects all the interesting places in the park. All bike paths are in parallel, meaning there are two lanes, one for biking in each direction.

As park popularity grows, NCPC decides to introduce electric four-wheel six-passenger wagon bikes for use within the park. However, these wagon bikes are so big that they cannot travel on a single bike lane. Instead, it must travel on both parallel bike lanes. Therefore, all parallel bike lanes must be changed into one-way bike lane. Direction of each road segment (directly connecting two interesting places) can be determined independently.

Due to popular demand, some places must be reachable from some other places. Give this set of must-reachable places, please help determine different ways to turn the parallel bike lanes into one-way wagon bike lane while meeting the must-reachable places conditions.

For example, in the Fig. 1 below (this figure depicts the Sample Input), there are 6 parallel bike lanes connecting 5 interesting places. If the park wants to maintain Place 4 to be reachable from Place 2, and Place 1 to be reachable from Place 5. Then there are 4 possible configurations as shown in Fig 2. It can be seen that only segment 3 and segment 5 must always set to have traffic flow in the same direction regardless of the configuration; while the other segments, may need to be set to different direction depending on the configuration.

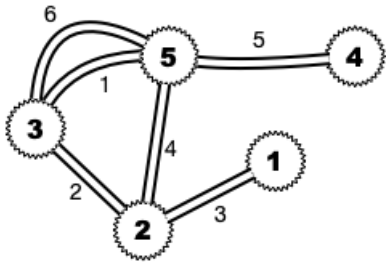


Figure 1: 6 Parallel bike lanes connecting 5 places.

Input File Format

The first line of input is an integer indicating the number of test cases to follow. For each test case, the first line contains 3 integers, *n, m, k*, indicating there are $n \leq 100,000$ interesting places in the park (numbered from 1 to n); there are $m \leq 100,000$ road segments (numbered from 1 to m); and there are $k \leq 100$ must-reachable place pairs.

The next m lines each contains two positive integers, $1 \leq i, j \leq n$, indicating there is road segment (parallel bike lanes) connecting Place i and Place j.

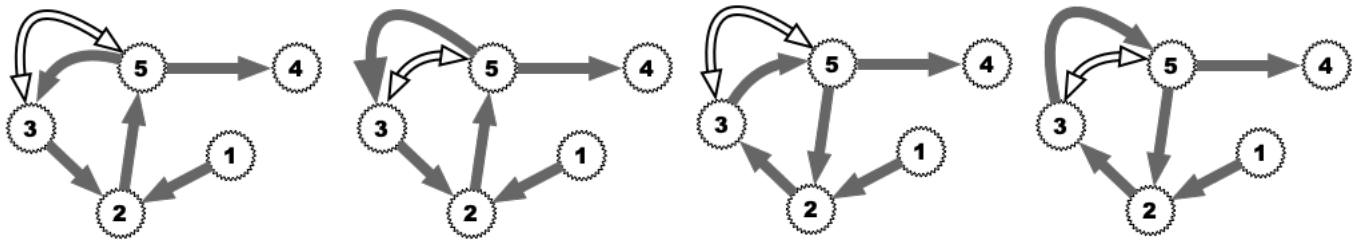


Figure 2: Four possible configurations. Segment 3 direction must be from Place 1 to Place 2 and segment 5 direction must be from Place 5 to Place 4 in all configurations. Segment with double arrows can be set in any direction in that particular configuration.

The next k lines each contains two positive integers, $1 \leq i, j \leq n$, indicating Place j must be reachable from Place i in the new configuration.

Output Format

Print "0", "L", or "R" for each road segment in the order of the input sequence, all on one line. For a road segment that connects Places i and j (as indicated in the input), if in all valid configurations, the direction is always from i to j, then output "R"; If the direction is always from j to i, then output "L"; otherwise, output "0". For each test case, there should be m consecutive "0", "L", or "R" characters on a single output line.

Sample Input

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

Sample Output for the Sample Input

```
00R0L0
```

Note: Road segment 3 (from place P1 to place P2), the direction should be "R" as in P1→P2. For road segment 5 (from place P4 to place P5), the direction must be "L", as in P4←P5).

Problem G
Viral Test
Time limit: 1 second

Dr. Smith accidentally discovers that the blood glucose level seems to be related to the incidence of a new virus. He starts collecting data from his patients, including blood glucose level and test results (positive or negative), as shown in the table below.

patient id	blood glucose level	test result
1	125	**positive**
2	100	**positive**
3	70	negative
4	120	**positive**
5	95	**positive**
6	60	negative
7	220	**positive**
8	85	negative
9	75	**positive**
10	90	negative

Dr. Smith believes that the results of the viral tests can be predicted based on blood glucose levels. The prediction procedure is simple: if the blood glucose level is greater than a threshold, a viral infection is detected (positive); otherwise no infection is found (negative) and vice versa.

Taking data displayed in the above table as an example, the best threshold value falls between 90 and 95, because the blood glucose levels with negative test results are all lower than or equal to 90, and those with positive test results are higher or equal to 95, except for one (patient id: 9). Therefore, the prediction accuracy rate is 90% (9/10), which is the best prediction result of this prediction procedure. Given patients' data, please write a program to help Dr. Smith determine the optimal threshold. The output of your program should be the number of samples which are correctly predicted.

Input File Format

The test data file may contain many test cases. Each test case contains several lines. The first line contains an integer N ($1 < N < 10^5$), indicating the number of samples. Each of the following N lines contains one real number b ($60.0 < b < 300.0$) and one binary number t, indicating the blood glucose level and test result (1: positive, 0: negative). The blood glucose levels in each test case are distinct. The last test case is followed by a line containing a single 0.

Output Format

The output for each test case is the number of correctly predicted samples in a best prediction result.

Sample Input

```
4
80.5 1
90.5 0
100.5 0
110.5 0
10
125 1
100 1
70 0
120 1
95 1
60 0
220 1
85 0
75 1
90 0
0
```

Sample Output for the Sample Input

```
4
9
```

Problem H
Robot Communication Cost
Time limit: 1 second

Assume there are N robots indexed from 1 to N. Specifically, robot 1 and robot N are stationary, and the other can move freely. Joe is the project manager, and he has M communication records among the robots, each of which has three positive integers: i, j, t indicating robot i initiates a communication with robot j for t seconds during a period of time, and there will be a charge on robot i, defined by a linear integral cost function $c(t) = at + b$. Unfortunately, except robot 1 and robot N, there is no information about the location when the robots communicated. To understand the impact of the distribution of robots, Joe tries to properly associate the rest $N - 2$ robots with robot 1 or robot N. Joe partitions the robots into two groups X and Y, one with robot 1 and the rest with robot N. Let L be the total charge from X to Y and R be the total charge from Y to X. With a proper partition Joe wants to minimize $L - R$. Your task is to write a program to help Joe find the minimum possibility.

Technical Specification

1. $1 < N < 500$
2. $1 < M < 5,000$
3. $0 < t < 1,000$
4. $-10 < a < 10,\ 0 \leq b < 1,000$

Input File Format

The first line of the input gives the number of test cases, T (< 10). For each case, the first line consists of two positive integers N, M indicating the number of robots and the number of communications made, respectively. The second line consists of two integers a, b indicating the cost function $c(x) = ax + b$. Then M lines follow, where the k-th ($k = 1, \ldots, M$) line has 3 positive integer i, j, t, indicating robot i communicates with j for t seconds.

Output Format

For each test case, output one line that contains the minimum value.

Sample Input

```
2
4 3
1 0
1 2 1
2 3 5
3 4 3
5 5
-1 10
1 2 2
2 3 3
1 3 2
2 4 1
5 2 4
```

Sample Output for the Sample Input

```
-1
-8
```

Problem I
Heaven's Coin
Time limit: 3 seconds

Believe it or not, strings can save creatures to heaven. One day Angel Tenshi found a way to save creatures to heaven. The secret is as follows. Every creature (including Tenshi) is equipped with a unique string that records the goodness and evil of this creature. This string is called the *karma*. (Of course, Tenshi's karma only has goodness.) Let us see how Tenshi can save a creature A to heaven. Let B be either Tenshi or a creature that was previously saved by Tenshi. Tenshi can save A if there is a way to match a suffix of B's karma to a prefix of A's karma, and in addition to this Tenshi has to spend some amount of *Heaven's Coin* in order to save A. (A match means that these two strings are exactly identical.) The amount of Heaven's coin that Tenshi needs for saving A is equal to the number of unmatched characters in A's karma. Notice that the saving order of creatures can drastically affect the total spend of Heaven's coin for Tenshi. Can you help Tenshi to find the minimum amount of Heaven's coin in order to save n creatures? For simplicity, we will use lowercase English letters as the characters of karmas. We use S_i to denote the karma of the ith creature, which is a finite-length string. For example, let the karma of Tenshi be `aaa`. Suppose there are 3 creatures, namely $S_1 =$ `aab`, $S_2 =$ `baa`, and $S_3 =$ `cba`, that Tenshi wants to save. If Tenshi applies the greedy strategy to save them, Tenshi would first save S_1 with cost 1. Then Tenshi will save S_2 with cost 2 because there is a length-1 suffix-to-prefix overlap between S_1 and S_2. Finally, Tehshi would save S_3 with cost 3. The total cost is $1 + 2 + 3 = 6$. However, if Tenshi follows this order S_1, S_3 and S_2, the cost for individual saving would be 1, 3 and 1, respectively. Now the total cost becomes $1 + 3 + 1 = 5$, which is better than the result of the greedy strategy. This moral tells us that greed cannot open the door to heaven.

Input Format

The test data file contains at most 10 test cases. The first line of the input gives you the number of test cases. In each test case, its first line specifies the integer n, which is the number of creatures that Tenshi needs to save, and $1 \leq n \leq 500$. Its second line is Tenshi's karma, and its third to the $(n + 2)$nd lines are the karmas of these creatures. A karma is a nonempty string of English lowercase letters and its length is at most 500 characters. The next test case follows immediately after the previous one.

Output Format

The output for each test case is the minimum amount of Heaven's coin that Tenshi needs in order to save the specified n creatures.

Sample Input

```
3
1
aaaaa
aabbb
2
ababab
abab
ababa
2
aaaa
bbbbb
ccc
```

Sample Output for the Sample Input

```
3
1
8
```

Problem J
Robot Dispatch Problem
Time limit: 2 seconds

Let us consider a sensing field on which sensors are deployed. These sensors will monitor their surroundings and report where events occur. For convenience, the occurrence of each event is described as one point in the sensing field (e.g., the center of the sensors that detect the event). Furthermore, there also exist a number of robots arbitrarily distributed in the sensing field. When some events occur, robots will be asked to move to the locations of these events to do analysis and reaction. To improve the efficiency of dispatch, each event location should be visited by exactly one robot. The sensing field contains no obstacles, so a robot is free to move to any position in the sensing field. However, since events may not last for a long time, robots should move to event locations as fast as possible. Thus, once an event location is assigned to a robot, this robot will move straight to that event location.

The robot dispatch problem is formulated as follows:

> Suppose that a set of distinct event locations $\hat{\mathcal{L}} = \{l_1, l_2, \cdots, l_n\}$ are reported by sensors in the sensing field. We are given a set of robots $\hat{\mathcal{R}} = \{r_1, r_2, \cdots, r_n\}$ used to analyze events. Let $d(r_i, l_j)$ be the Euclidean distance between the current position of a robot $r_i \in \hat{\mathcal{R}}$ and an event location $l_j \in \hat{\mathcal{L}}$, where $d(r_i, l_j) \geq 0$. For each event location l_j in $\hat{\mathcal{L}}$, this problem asks how to assign one robot to visit it, such that the total moving distance of robots in $\hat{\mathcal{R}}$ can be minimized.

Please write a program to calculate the best assignment of robots. You will be given the distance between each event location and every robot (in a 2D array). The output of your program should be the overall moving distance of robots based on your assignment.

Technical Specification

1. There are 10 test cases.
2. $10 \leq n \leq 50$.
3. The value of each $d(r_i, l_j)$ item must be a positive integer.
4. $10 < d(r_i, l_j) < 100$.

Input Format

Each test case contains an integer n followed by an $n \times n$ array, where $n > 1$. The (i, j)-th element in the array is the distance $d(r_i, l_j)$ between robot r_i and event location l_j. The last test case is followed by a line containing a single 0.

Output Format

The output for each test case is the total moving distance of robots.

Sample Input

```
2
3 7
8 9
3
40 60 15
25 30 45
55 30 25
0
```

Sample Output for the Sample Input

```
12
70
```

Problem K
Climbing Stairs
Time limit: 3 seconds

There is a long staircase over a mountain. It takes N-steps stairs to reach the top. Each time you can climb 1, 2, 3, $\cdots$, or H steps. But if you don't want to climb all the steps, you can hire bearers at 0^{th} step to take you to some k^{th} step and then climb to the top by yourself. But the bearers will charge you $0, 1, 2, \cdots$, or $k^H \times H^k$ dollars randomly by rolling a wheel. How many distinct methods(ANS) are there to reach the top? For the convenience of verifying the answer, please output ANS mod M for a given positive integer M.

Let us consider the example of $H = 2$, and $N = 3$. You can climb 1 step three times. The method is denoted as $(1, 1, 1)$. You can climb 1 step and then climb 2 steps, which is denoted as $(1, 2)$. You can climb 2 steps and then climb 1 step, which is denoted as $(2, 1)$. Bearers can take you to the first step, and then you climb 1 step twice, which is denoted as $(B1, 1, 1)$ with $1^2 \times 2^1 + 1$ methods. Bearers can take you to the first step, and then you climb 2 steps, which is denoted as $(B1, 2)$ with $1^2 \times 2^1 + 1$ methods. Bearers can take you to the second step, and then you climb 1 step, which is denoted as $(B2, 1)$ with $2^2 \times 2^2 + 1$ methods. Bearers can take you to the third step, which is denoted as $(B3)$ with $3^2 \times 2^3 + 1$ methods. So the number of methods are $1 + 1 + 1 + (1^2 \times 2^1 + 1) + (1^2 \times 2^1 + 1) + (2^2 \times 2^2 + 1) + (3^2 \times 2^3 + 1) = 99$. If an integer $M = 5$ is given, then please output $99 \bmod 5 = 4$.

Technical Specification

1. There are at most 10 test cases.
2. $2 \leq M \leq 1,000,000,000$.
3. $0 \leq k \leq N$
4. $1 \leq H \leq 15$.
5. $1 \leq N \leq 10^{64}$

Input Format

The first line contains an integer indicating the number of test cases. Each test case contains three integers, M, H, and N.

Output Format

For each test case, please output the number $(\bmod M)$ of distinct methods to reach the top.

Sample Input

```
2
999 2 3
8 1 3
```

Sample Output for the Sample Input

```
99
2
```

Problem L
Rescue Mission
Time limit: 4 seconds

The princess has been imprisoned in the bottom-right corner of a castle. The castle consists of ***M* x *N*** rooms laid out in a 2D grid. Mario was initially positioned in the top-left room and must fight his way through the castle to rescue the princess. In order to reach the princess as quickly as possible, Mario decides to move only rightward or downward in each step.

Mario has an initial health point represented by a positive integer. If at any point his health point drops to 0 or below, he dies immediately.

Some rooms are guarded by monsters, so Mario loses health (*negative integer*) upon entering these rooms; some rooms are either empty (*0's*) or contain magic mushrooms that increase Mario's health (*positive integers*); the other rooms are blocked that Mario couldn't pass through.

Please write a computer program to determine Mario's minimum initial health point so that he is able to rescue the princess. If Mario cannot complete the mission due to blocked rooms, return -1.

Note:

- Any room can contain monsters or mushrooms, even the first room Mario enters and the bottom-right room where the princess is imprisoned.

Technical Specification

1. Mario's health point is at most 10^9.
2. $2 \leq M \leq 1000$, M is an integer.
3. $2 \leq N \leq 1000$, N is an integer.
4. Health points gained/lost from a room are integers.
5. If a room is blocked, Mario can't enter this room and the value of this room is -1001.
6. And if a room is not blocked, $-1000 \leq$ the value of this room ≤ 1000

Input File Format

The test data file contains many test cases. The first line gives you the number of test cases.

The second line contains two integers, indicating $\boldsymbol{M}$, $\boldsymbol{N}$ of case 1 respectively.

For the next $\boldsymbol{M}$ lines, each line has $\boldsymbol{N}$ integers. Each integer represents a room in the castle and the health points should be gained/lost from this room. If the integer is -1001, it implies this room is blocked.

The next line contains two integers, indicating $\boldsymbol{M}$, $\boldsymbol{N}$ of case 2 respectively, and so on.

Output Format

The output of each test case is either an positive integer or -1. Each output should be placed in a separate line.

Sample Input

```
2
2 2
-10 -1001
-1001 -10
3 3
-200 -300 -1001
-500 -1000 100
1000 -1001 -500
```

Output for the Sample Input

```
-1
1901
```

Problem M
The Summit from Where I Stand
Time limit: 1 second

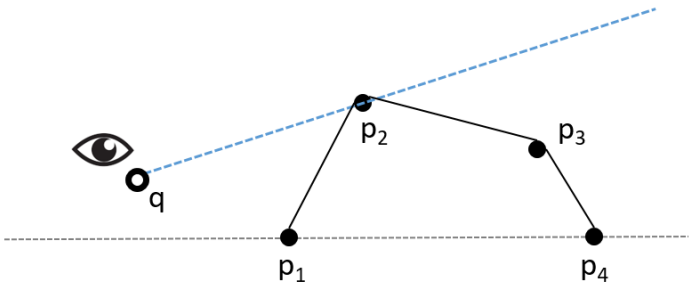
Ivan loves climbing mountains. When he is on his hiking trip, he often looks into the distance and observes the peaks of the surrounding mountains. He found an interesting fact that, when he looks towards the same mountain from different viewpoints, the "uppermost" point of the mountain he can see varies with the position he currently stands in.

To simplify the scenario, assume that Ivan's position is described by a point $\mathbf{q} = (x_q, y_q)$ in the second quadrant of the 2-D plane and the shape of the mountain is described by a convex polygonal curve $\mathbf{p}_1 = (x_1, y_1), \mathbf{p}_2 = (x_2, y_2), \ldots, \mathbf{p}_n = (x_n, y_n)$ in the first quadrant of the plane which begins and ends both at the x-axis.

You can further assume that

- $0 < |x_q| \leq 10^5$, $|y_q| \leq 10^8$,
- $x_1, x_2, \ldots, x_n$ is non-decreasing, and
- $\max\limits_{1 \leq i \leq n} \{|x_i|, |y_i|\} \leq 10^8$.

Given Ivan's position and the curve of the mountain he is observing, the uppermost point of the mountain, or, the summit point, Ivan sees from his current position is defined to be the upper tangent point of his eyesight towards the curve of the mountain. The uppermost point is defined to be $\mathbf{p}_n$ if no such tangent point exists.



In this problem you are to verify Ivan's intriguing observation. Given the curve of the mountain and a set of m viewpoints from which Ivan has observed that mountain during his hiking trip, please output the index of the summit point he sees at each viewpoint.

Input Format

The input consists of multiple testcases. Each testcase starts with a line containing two integers n and m, where $3 \leq n \leq 10^5$ and $1 \leq m \leq 10^5$.

Then there are n lines, each of which contains two integers x_i and y_i as described above. After that there are m lines, one for each query viewpoint. Each of these m lines contains two integers which are the coordinates of each query viewpoint.

A testcase starting with $n = m = 0$ indicates the end of input.

Output Format

For each viewpoint in each testcase of the input, print the index of the summit point Ivan sees in a line. If there are multiple solutions, print the one with the smallest index.

Sample Input

```
3 4
0 0
0 10
10 0
-1 0
-1 5
-1 11
-1 12
3 1
0 0
1 1
2 0
-2 1
0 0
```

Sample Output for the Sample Input

```
2
2
2
3
2
```

Problem N
Server Connectivity
Time limit: 10 seconds

For a graph $G = (V, E)$ and a subset $S \subseteq V$, we use $G[S]$ to denote the subgraph of G *induced* by S, which is the graph obtained from G by removing all vertices not in S and their incident edges. For instance, for the graph $G = (V, E)$ in Figure 1, where $V = \{1, 2, ..., 7\}$ and $E = \{(7, 2), (1, 3), (2, 4), (5, 1), (2, 6)\}$ and $S = \{2, 4, 7\}$, we have $G[S] = (V', E')$, where $V' = \{2, 4, 7\}$ and $E' = \{(7, 2), (2, 4)\}$ (See Figure 2). For a graph $G = (V, E)$, we say that G is *connected* if there is a path between every pair of vertices $u, v \in V$. For example, the graph in Figure 1 is not connected, while the graph in Figure 2 is connected.

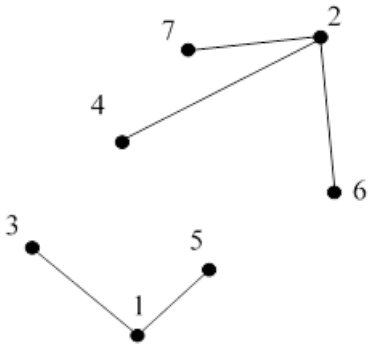


Figure 3: G

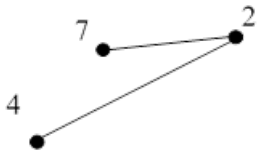
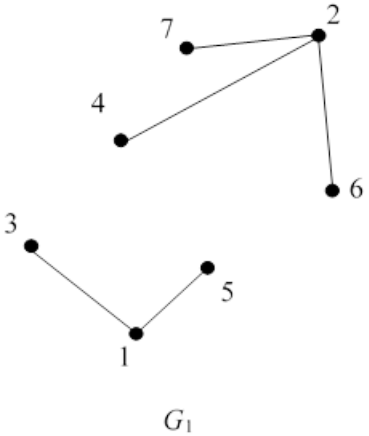
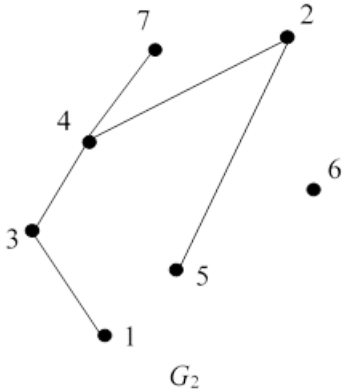


Figure 4: $G[S]$



G_1



G_2

Figure 5: G_1 and G_2

A software company has a set of servers around the world to handle their global service. The servers are linked to each other by two types of connections, where connections of each type form a network that contains no cycles. These two networks are represented by two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$, where V is the set of servers and E_1 (E_2) is the

set of connections of type 1 (type 2). The owner wishes to solve a problem described as follows. A subset $S \subseteq V$ is a *common connected component* if $G_1[S]$ and $G_2[S]$ are both connected. For instance, in Figure 3, $S = \{2, 4, 7\}$ is a common connected component of G_1 and G_2, but $S' = \{4, 7\}$ is not. A common connected component S is *maximal* if it is not contained in a larger common connected component. For example, in Figure 3, $S = \{2, 4, 7\}$ and $S'' = \{2, 4\}$ are both common connected components. In this example, S is maximal, but S'' is not. For the example in Figure 3, the set of maximal components is {{2, 7, 4}, {1, 3}, {5}, {6}}. It is easy to observe that the set of maximal common connected components is unique and represents a partition of the vertex set.

Please write a program to find the number of maximal common connected components and the size of the largest common connected component of two given graphs G_1 and G_2.

Technical Specification

- The number of test cases is at most 25.
- The number, n, of vertices is an integer between 1 and 2×10^5.
- The vertex set is $V = \{1, 2, ..., n\}$.
- G_1 and G_2 are (undirected) graphs with no cycles.

Input File Format

The first line of the input is an integer t, indicating there are t test cases. The first line of each test case gives 3 integers n, m_1, and m_2, where n is the number of vertices and m_1 (m_2) is the number of edges in G_1 (G_2). Then, $m_1 + m_2$ lines follow, where the first m_1 lines give the edges of G_1, the remaining lines give the edges of G_2, and each line contains two integers i and j ($1 \leq i, j \leq n$), indicating there is an edge connecting vertex i and vertex j.

Output Format

For each case, output two integers in a line, where the first is the number of maximal common connected component and the second is the size of the largest common connected component. For the example in Figure 3, the output is 4 and 3.

Sample Input

```
3
7 5 5
7 2
2 4
1 3
2 6
5 1
3 4
2 5
3 1
4 2
7 4
3 2 2
1 2
2 3
1 3
3 2
4 3 2
1 2
1 4
1 3
3 4
2 4
```

Output for the Sample Input

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