Problem A
Looping Cycle
Max no. of test cases: 20
Time limit: 1 second

The "looping cycle" of a number is a subsequence of outcomes when repeatedly calculating the sum of power of digits of the number. Specifically, for a given positive integer N_0, we can calculate the sum of power of digits of N_0 and obtained a new integer N_1. Then, we can repeat the process to obtain the sequence N_2, N_3, N_4, ..., and so on.

During the process, there must exist a subsequence (N_i, N_{i+1}, N_{i+2}, ..., , N_{i+k}) that repeatedly back-to-back appear in the sequence of outcomes when repeatedly calculating the sum of power of digits of the given number N_0. This subsequence is called the "looping cycle" of N_0, and the length of this subsequence is $k + 1$.

For instance, suppose $N_0 = 5$, by repeatedly calculating the sum of power of digits of the number, we can obtain

$$N_1 = 5^2 = 25$$
$$N_2 = 2^2 + 5^2 = 29$$
$$N_3 = 2^2 + 9^2 = 85$$
$$N_4 = 8^2 + 5^2 = 89$$
$$N_5 = 8^2 + 9^2 = 145$$
$$N_6 = 1^2 + 4^2 + 5^2 = 42$$
$$N_7 = 4^2 + 2^2 = 20$$
$$N_8 = 2^2 + 0^2 = 4$$
$$N_9 = 4^2 = 16$$
$$N_{10} = 1^2 + 6^2 = 37$$
$$N_{11} = 3^2 + 7^2 = 58$$
$$N_{12} = 5^2 + 8^2 = 89$$
$$N_{13} = 8^2 + 9^2 = 145$$
$$...$$

Thus, it is obvious that (89, 145, 42, 20, 4, 16, 37, 58) is the "looping cycle" of the number 5, and the length of 5's "looping cycle" is 8.

Now, your task is to find the length of the "looping cycle" for the given positive integers.

Input File Format

The first line has an integer denoting the number of test cases to follow. For each test case, there is one positive integer N_0 in a line, and, for simplicity, N_0 is less than 1,000,000.

Output Format

For each test case, please output the length of the corresponding looping cycle in a line.

Sample Input

```
5
49
1
11
```

Output for the Sample Input

```
8
1
1
8
```

Problem B
Traveling Number
Max no. of test cases: 15
Time limit: 1 second

Let's define a trip of a positive integer by the following steps:

1. Starts the trip with the leftmost digit of the number.
2. Continues the trip by moving to the right, wrapping around to the left end of the number if necessary, as many digits as the value of the currently visited digit.
3. Stop the trip if the current digit has been visited before; otherwise, repeat Step 2.

Then, a number is called a 'Traveling Number', if all the digits of the number can be visited in a trip starting from its leftmost digit.

For instance, 31742 is a valid traveling number, because the trip starts with the leftmost digit 3 and visits all other digits 4, 7, 2, 1 in order. However, 31724 is NOT a traveling number, because the trip starts with the leftmost digit 3, and then visits the 4th digit 2. Then, it goes to the leftmost digit 3 again and finishes the trip by rule without visiting all the digits in the journey.

Your task is to write a program and tell whether the input number is a valid traveling number.

Input File Format

The first line has an integer denoting the number of test cases to follow. For each test case, there is one positive integer N in a line. For simplicity, N is less than 1,000,000,000, and each digit number is an integer from 0 to 9 (inclusive).

Output Format

For each test case, please output whether the input number is a traveling number in a line. Specifically, please output "Yes" if the input number is a valid traveling number; and output "No" otherwise.

Sample Input

```
31742
31724
111
12
```

Output for the Sample Input

```
Yes
No
Yes
Yes
```

Problem C
Invest in Stocks
Max no. of test cases: 10
Time limit: 1 second

Suppose that you have been offered an opportunity to invest in the Volatile Chemical Corporation. Like the chemicals the company produces, the stock price of the Volatile Chemical Corporation is rather volatile. You are allowed to buy one unit of stock and then sell it at a later date, buying and selling after the close of trading for the day. After you sell the stock, you can buy it again. You can do it several times. However, at any time, you can hold at most one unit of the stock. To compensate for this restriction, you are informed the future prices of the stock. Your goal is to maximize your profit.

For example, the following table shows stock prices over a 6-day period. You may buy the stock on day 2 and sell it on day 4, buy it again on day 5 and sell it on day 6. The profit is 9, which is the highest among all strategies. Certainly you do not have to buy the stocks. In the second example (please see sample input 2), the stock prices are decreasing, therefore the best strategy is do nothing. You do not have any profit if you hold the stock but do not sell it.

day	1	2	3	4	5	6
price	112	111	113	117	115	118

Please write a program to find the maximum profit, given a sequence of stock prices.

Input File Format

The first line has an integer denoting the number of test cases to follow. Each test case has two lines. The first line contains one integer N ($1 \leq N \leq 10^4$), indicating the number of stock prices. The second line contains N integers, $p_1, p_2, \ldots, p_N$ ($1 \leq p_i \leq 10^9$), indicating the stock prices.

Output Format

For each test case, please print the maximum profit.

Sample Input

```
2
6
112 111 113 117 115 118
5
210 209 208 207 206
```

Output for the Sample Input

```
9
0
```

Problem D
Maximum Profit Interval
Max no. of test cases: 10
Time limit: 1 second

Professor Oram is trying to invest his savings in the stock market. He is not very good at stock market trading, and he is not very lucky. Recently, every time he buys a stock, the price of that stock starts to fall, and every time he sells a stock, the price of that stock starts to rise. So he decided to study the stock market, trying to understand the best times to buy and sell stocks from past market data.

Professor Oram decided to invest in each stock for the long term. That is, each stock is traded only once: buy it once and sell it later. He wants to know, for each stock, when is the best time to buy and the best time to sell the stock for maximum profit or minimum loss.

Professor Oram has collected a large amount of price data for many stocks. To save storage space, the price of a stock at time $t = 0, 1, \ldots, n$

$$p_0, p_1, \ldots, p_n$$

will be stored as

$$x_1, x_2, \ldots, x_n$$

where $x_i = p_i - p_{i-1}$. Since the initial price of the stock p_0 is not important to the study, it will not be stored. The following is an example of a stock market data for $n = 8$.

	$t=0$	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$
p_t	1000	1005	1020	990	1000	995	1035	1045	1040
x_t		5	15	-30	10	-5	40	10	-5

These data will be stored as 8 integers.

$$5, 15, -30, 10, -5, 40, 10, -5$$

In the above example, the best time to buy the stock is $t = 3$, the best time to sell the stock is $t = 7$, and the profit will be 55. Note that the sale time must be after the purchase time. They cannot occur at the same time.

Write an efficient program for Professor Oram that can process large amounts of stock market data he collects.

Input File Format

For testing purposes, the stock market data for this problem will be randomly generated rather than read from an actual market price file.

There are more than one test cases in the input file. Each test case starts with a positive integer n, and then followed by four positive integers a, b, c, d. The n prices data $x_1, x_2, \ldots, x_n$ are generated from these four integers as follows.

1. First generate the sequence $y_0, y_1, \ldots, y_n$:

$$y_i = \begin{cases} 1 & \text{if } i = 0, \\ (ay_{i-1} + b) \bmod c & \text{if } i > 0. \end{cases}$$

2. Then let $x_i = (y_i \bmod d) - \lfloor d/2 \rfloor$ for $i = 1, 2, \ldots, n$.

In the above equations "$u \bmod v$" is the remainder of dividing u by v, and "$\lfloor d/2 \rfloor$" is the largest integer not exceeding $d/2$.

The value of n is at most 10^7, and the values of the other integers a, b, c, and d are bounded by 2^{31}.

The input data for each test case is given in one line, and the last line of the input file contains only an integer 0 to indicate the end of the input.

Output Format

For each test case, print the maximum profit P, the time to buy B, and the time to sale S in one line:

P B S

Print exactly one space between each data, and no other spaces.

If the interval with the maximum profit is not unique, print the interval with the smallest S. If there are more than 1 maximum profit ending with the same S, print the one with the largest B.

Sample Input

```
8 31 35 97 43
10 97 89 103 62
0
```

Output for the Sample Input

```
44 2 7
40 4 7
```

Problem E
Channel Sharing
Max no. of test cases: 5
Time limit: 1 second

Consider a communication network where many users share a single multiaccess channel. A user with a message to be transmitted is called an *active* user. For convenience, assume that each message is of unit length and can be transmitted in one time slot. The transmission is done by broadcasting over the channel which every user, *active* or not, can receive. However, if at any given time slot more than one active user broadcasts, then the messages conflict with each other and are reduced to noise. The problem is to design a schedule which arranges the transmissions of active users into different time slots so that the transmissions can be successful.

We consider the approaches here by scheduling in epochs. At the start of an epoch, the users are classified either as *active* or *inactive* according as whether they have messages to send at that particular moment. An inactive user remains labeled as inactive even though a message is generated during the epoch. An epoch ends when all active users have transmitted successfully in different time slots, and then the next epoch starts. If the set of active users is known, they can then be scheduled to transmit in successive time slots in some order and no conflict can occur . But active users are usually unknown beforehand. One simple protocol, called TDM (time-division-multiplexing), is to assign each user a time slot in which the user can transmit if active. When there are n users, n time slots will be needed. This is inefficient when the number of active users is typically small.

Table 1: Scheduling example with $n = 8, d = 3$

0	1	0	0	0	1	1	0	x/y
0	1	1	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	1	1
0	0	1	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	1	1	1
1	0	0	1	0	0	0	0	0

Suppose there are n users, numbered from 1 to n, sharing a common communication channel and for any epoch there are at most d active users. Let M be a scheduling matrix, where $M_{i,j} = 1$ if the j-th user can transmit message at the i-th time slot; 0 otherwise. Table 1 gives an example of the scheduling matrix with $n = 8$ and $d = 3$ together with the extra row x and column y. Row x, the first row, indicates the active users, i.e., $x_i = 1$ if user i is active; 0 otherwise. The rightmost column, labeled y, indicates if a corresponding active user successfully transmits message without collision.

Row x shows that users 2, 6 and 7 are active. The scheduling is define by the 7×8 matrix M, excluding row x and column y. The first row of the scheduling matrix, i.e., $(0, 1, 1, 0, 0, 0, 0, 0)$, means users 2 and 3 are legitimate to send message. Since at this time slot user 2 is the sole active user allowed to send message, user 2 successfully completes the transmission and y_1 is 1. The 2nd row of M shows that no active users are allowed to send message and thus $y_2 = 0$. It can be checked that these three active users can successfully transmit their message under this scheduling. But if the active users are user 1, 2 and 3, then this scheduling matrix fails.

Given n and d, your task is to write a program to find a scheduling matrix M such that no matter which d users are active, they can always successfully transmit their message.

Input File Format

The first line in the input file has a positive integer T, which indicates there are T (≤ 5) test cases. Then in each of the following T lines there are two positive integers, n (≤ 300) and d ($1 \leq d < 4, dn \leq 900$), which indicate, respectively, the number of users and upper bound of the number of active users for any epoch.

Output Format

For each test case, the first line should contain two integers, k and n, indicate, respectively, the scheduling matrix's row and column numbers. Then, in the following k lines, output a scheduling matrix row by row. The integer k should be as small as possible. Particularly, acceptable k should be at most $\min\{n, 2d^2 \log n\}$.

Sample Input

```
2
3 3
4 1
```

Output for the Sample Input

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

Problem F
Identifying Suspicious Samples
Max no. of test cases: 20
Time limit: 1 second

Ben is analyzing a dataset of n bacteria samples $S_1, S_2, ..., S_n$. Via a series of experiments, he classifies the relation between a pair of two samples S_i and S_j as follows:

(1) S_i and S_j have *ancestor-descendant relationship*;

(2) S_i and S_j are *independent*; and

(3) S_i and S_j are *in conflict.*

Here, "independent" indicates that S_i and S_j do not have ancestor-descendant relationship. If all samples are not corrupted, the relation of S_i and S_j should be either (1) or (2). An occurrence of relation (3) reveals that there are corrupted samples. Ben wants to identify corrupted samples. Based on his knowledge in biology, Ben devises a procedure to identify suspicious samples. Let us say that a sample S_i is *compatible at* a sample S_j if the following hold: (a) S_i is a descendant of S_j and (b) for all samples S_k such that $k \neq i$ and S_k is a descendant of S_j, S_i and S_k are not in conflict. A sample S_i is *regular* if there exists a sample S_j such that S_i is compatible at S_j. Ben's procedure goes in rounds. Each round consists of three steps:

Step 1: find all regular samples and mark them as *uncorrupted.*

Step 2: remove all samples marked in Step 1.

Step 3: if no sample is removed in Step 2, terminate. Otherwise, begin the next round.

After the procedure ends, all remaining samples are marked as *suspicious.* All suspicious samples will be further investigated to check whether they are corrupted. Please help Ben implement the above procedure.

Consider the example in Figure 1, in which an arrow from a sample S_i to a sample S_j indicates that S_i is a descendant of S_j, and a dashed line between two samples indicates that they are in conflict. There is no edge between a pair of independent samples. Since there are corrupted samples, the ancestor-descendant relationships may not be transitive. For example, in Figure 1, S_5 is a descendant of S_4 and S_4 is a descendant of S_2, but S_5 is not a descendant of S_2. In addition, the relationships may be cyclic. For example, in Figure 1, the ancestor-descendant relationships among S_5, S_4, and S_2 form a cycle. Ben's procedure goes as follows. At the beginning, the descendants of S_2 are S_3, S_4, and S_6. Since S_3 is not in conflict with S_4 and S_6, it is compatible at S_2 and thus is regular. Similarly, S_5 is compatible at S_3 (and at S_4.) and thus is regular. Both S_3 and S_5 are removed in Round 1. After the removal, S_7 and S_8 become regular and thus are removed in Round 2. No sample is removed in Round 3. Therefore, the procedure terminates at Round 3.

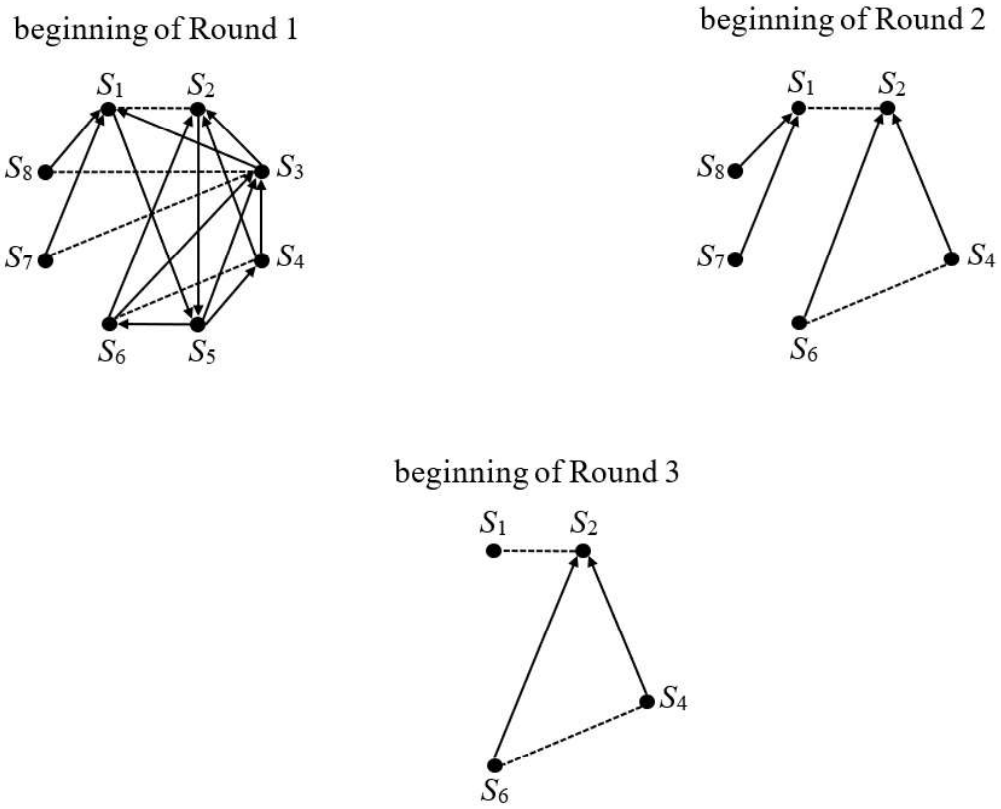


Figure 1. An example.

Technical Specification

1. The number of test cases is at most 20.
2. $1 \leq n \leq 5000$.
3. At most 2×10^5 pairs of samples have ancestor-descendant relationships.
4. At most 2×10^5 pairs of samples are in conflict.

Input File Format

The first line of the input contains an integer indicating the number of test cases. Each test case begins with a line containing three integers n, b, and c. The integers b and c indicate that there are b pairs of samples which have ancestor-descendant relationships, c pairs of samples which are in conflict, and $n \times (n-1)/2 - b - c$ pairs of samples which are independent. The next b lines specify the ancestor-descendant relationships: Each line contains two integers x and y, indicating that S_x is a descendant of S_y (and S_y is an ancestor of S_x). Each of the next c lines contains two integers x and y, indicating that S_x and S_y are in conflict.

Output Format

For each test case, print a line containing two integers d and k, where d indicates that the procedure terminates at the d-th round and k is the number of suspicious samples.

Sample Input

```
4
8 13 4
8 1
7 1
3 1
3 2
4 2
6 2
4 3
5 3
6 3
1 5
2 5
5 4
5 6
1 2
8 3
7 3
6 4
3 2 0
2 1
3 2
4 0 0
3 3 0
1 2
2 3
3 1
```

Output for the Sample Input

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

Problem G
Rounding
Max no. of test cases: 10
Time limit: 1 second

Professor X is preparing a simple task for NCPC. The input of the task is an n by n matrix of integers and floating point numbers. The contestants are asked to compute the sums of each row and each column.

Right after Professor X prepared the dataset, he asks his assistant, JUNIOR Y, to verify the dataset. Unfortunately, JUNIOR Y does not know how to manipulate floating point numbers. Professor X is then aware that the task is too difficult and decides to simplify the task by using integers instead of floating point numbers. However, he does not want to regenerate the dataset and decides to modify the inputs and outputs as follows. Let $a_{i,j}$ be the entry at row i and column j of an input matrix, and $b_{i,j}$ be the rounded result. Then

- For $i, j \in \{1, 2, \ldots, n\}$, $b_{i,j} = \lfloor a_{i,j} \rfloor$ or $b_{i,j} = \lceil a_{i,j} \rceil$.
- For $i \in \{1, 2, \ldots, n\}$, $\displaystyle\sum_{j=1}^{n} b_{i,j} = \left\lfloor \sum_{j=1}^{n} a_{i,j} \right\rfloor$ or $\displaystyle\sum_{j=1}^{n} b_{i,j} = \left\lceil \sum_{j=1}^{n} a_{i,j} \right\rceil$.
- For $j \in \{1, 2, \ldots, n\}$, $\displaystyle\sum_{i=1}^{n} b_{i,j} = \left\lfloor \sum_{i=1}^{n} a_{i,j} \right\rfloor$ or $\displaystyle\sum_{i=1}^{n} b_{i,j} = \left\lceil \sum_{i=1}^{n} a_{i,j} \right\rceil$.

It is known that each column and each row in the original input matrix contains at most two non-integer entries. Please write a program to help Professor X to see if the requested modification is possible, and give a modified matrix if it exists.

For example, consider the following matrix.

$$\begin{pmatrix} 6.3 & 7 & 4.6 \\ 4.7 & 2 & 2 \\ 5 & 4.5 & 3.4 \end{pmatrix}$$

The sums of the rows are 17.9, 8.7, and 12.9, respectively, and those of the columns are 16, 13.5, and 10, respectively. The requested modification is possible. One feasible modification is

$$\begin{pmatrix} 7 & 7 & 4 \\ 4 & 2 & 2 \\ 5 & 4 & 4 \end{pmatrix}$$

whose sums of rows are 18, 8, and 13, respectively, and the sums of columns are 16, 13, and 10, respectively.

Input File Format

The first line of the input is an integer, specifying the number of test cases. For each test case, the first line contains an integer n ($2 \leq n \leq 500$), which is the number of rows/columns of the matrix. Each of the following n lines is a row of the matrix, containing n positive numbers. Every number is either an integer or a floating number, where a floating point number is represented to one place of decimal. Every number in the matrix is at most 100. Two consecutive numbers in a line are separated by a space.

Output Format

For each test case, output the requested integer matrix if the requested modification is possible. The output consists of n lines, where the ith line corresponds to the ith row of the integer matrix. Each line contains n integers, and two consecutive integers in a line are separated by a space. If the requested integer matrix does not exist, output -1.

Sample Input

```
2
2
0.5 0.6
0.7 0.9
3
6.3 7 4.6
4.7 2 2
5 4.5 3.4
```

Output for the Sample Input

```
0 1
1 1
7 7 4
4 2 2
5 4 4
```

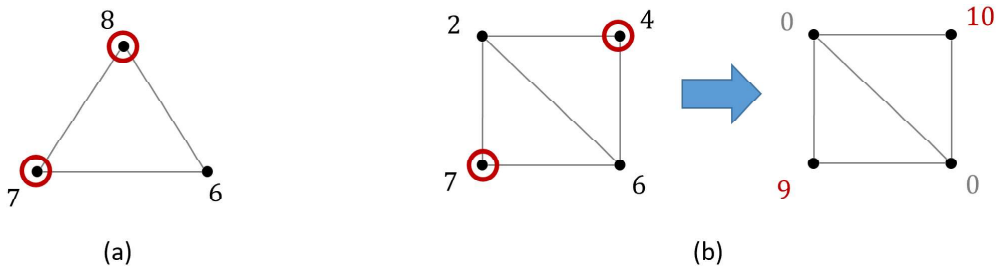
Problem H
Organize the Rocks
Max no. of test cases: 15
Time limit: 2 seconds

A number of rocks are scattered here and there on the vertices of a connected graph $G = (V, E)$, waiting to be organized tidily. In this problem, your task is to decide, whether or not a given plan to organizing the rocks is possible to be carried out.

The scenario is as follows. Each vertex $u \in V$ in the graph has w_u units of heavy rocks, that can be moved freely to other vertices that are not too far away. Precisely speaking, the rocks located at a vertex u can be moved to another vertex v only if $d(u, v) \leq X$, i.e., the distance between u and v in G is at most X, where $d(u, v)$ is the length of any shortest path between u and v in G.

The problem setter has a subset A of vertices to which he would like to collect the rocks. As the rocks are so heavy, each of these vertices can accommodate at most M units of rocks. To be precise, the problem setter would like to move the rocks to the vertices in A, such that any vertex in A has at most M units of rocks and any $v \notin A$ has no rocks at all. Please decide for the problem setter if his plan is possible to be carried out.

For example, consider the following two graphs given in (a) and (b), where the grey hollow circles denote the vertices in A. Suppose that $M = 10$ and $X = 3$. Then, the given plan in (a) is not possible while the given plan in (b) is possible, as shown below.



Input File Format

The first line of input contains one integer T, $1 \leq T \leq 15$, denoting the number of test cases to follow.

Each test case starts with four integers n, m, M, and X in a line, where $1 \leq n \leq 10^3$, $1 \leq m \leq 10^5$, $1 \leq M \leq 10$, and $1 \leq X \leq n-1$, that stand for the number of vertices, the number of edges in G, the maximum units of rocks a vertex can accommodate, and the

maximum distance for which the rocks can be moved. The vertices are numbered from 1 to n.

The next line contains n integers, i.e., $w_1, w_2, \ldots, w_n$, the number of rocks each vertex has. You may assume that $0 \leq w_v \leq M$ for each $1 \leq v \leq n$.

Each of the following m lines contains two integers u and v, indicating that there is an edge between u and v.

The next line starts with the integer $|A|$, the size of A, followed by $|A|$ integers, the vertices in A.

Output Format

For each test case, if the given plan is possible, print "Yes" in a line.
Otherwise, print "No" in a line.

Sample Input

```
2
3 3 10 2
8 7 6
1 2
2 3
3 1
2 1 2
4 5 10 3
2 7 6 4
1 2
2 3
3 4
4 1
3 1
2 2 4
```

Output for the Sample Input

```
No
Yes
```

Problem I
Approximating Distances
Max no. of test cases: 5
Time limit: 2 seconds

The NCPC queen has n points, denoted 0, 1, ..., $n-1$, and a positive integer $3 \leq h \leq 10$. The distance between any distinct points i and j is

$$d(i,j) = \begin{cases} |i-j|+2, & \text{if } (i \bmod 3) = 0 \text{ and } (j \bmod 5) = 1, \\ |i-j|+2, & \text{if } (j \bmod 3) = 0 \text{ and } (i \bmod 5) = 1, \\ |i-j|+1, & \text{otherwise.} \end{cases}$$

For each point i, $d(i,i) = 0$. For integers $a \geq 0$ and $b \geq 1$, denote by $a \bmod b$ the remainder of the division of a by b.

Let $t = \lceil n^{1/h} \rceil$, which is the least integer greater than or equal to $n^{1/h}$. For all i, $j \in \{0, 1, \ldots, n-1\}$, the queen approximates $d(i, (i+j) \bmod n)$ by

$$\tilde{d}(i, (i+j) \bmod n) = \sum_{k=0}^{h-1} d\left(\left(i + \sum_{\ell=h-k}^{h-1} s_\ell(j) \cdot t^\ell \right) \bmod n,\ \left(i + \sum_{\ell=h-1-k}^{h-1} s_\ell(j) \cdot t^\ell \right) \bmod n \right),$$

where $(s_{h-1}(j), s_{h-2}(j), \ldots, s_0(j))$ denotes the t-ary representation of j (i.e., $s_{h-1}(j)$, $s_{h-2}(j)$, ..., $s_0(j) \in \{0, 1, \ldots, t-1\}$ and $\sum_{\ell=0}^{h-1} s_\ell(j) \cdot t^\ell = j$). By convention, empty sums vanish, e.g., $\sum_{\ell=h}^{h-1} s_\ell(j) \cdot t^\ell = 0$. With the queen's approximation for $d(i, (i+j) \bmod n)$, we can now approximate the total distance from i to all points by

$$\sum_{j=0}^{n-1} \tilde{d}(i, (i+j) \bmod n).$$

Please find the remainder of the division of this quantity by 257, for all $i \in \{0, 1, \ldots, n-1\}$. Note that $\Omega(n^2)$-time algorithms may exceed the time limit.

Input File Format

The first line has an integer denoting the number of test cases to follow. For each test case, there are two integers on a single line, namely n and h (in that order), where $3 \leq n \leq 100000$ and $3 \leq h \leq 10$. It is guaranteed that $n^{1/h}$ is not an integer.

Output Format

For each test case and $i = 0$, 1 up to $n - 1$, output

$$\left(\sum_{j=0}^{n-1} \tilde{d}\left(i, (i+j) \bmod n\right) \right) \bmod 257.$$

Separate two consecutive numbers by the newline character.

Sample Input

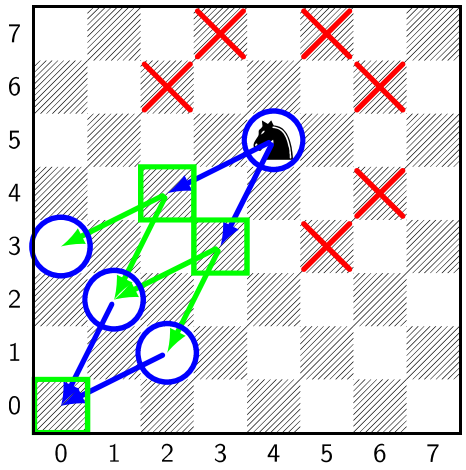
```
2
5 4
6 3
```

Output for the Sample Input

```
16
15
15
15
17
23
28
19
24
23
26
```

Problem J
Pony Knight
Max no. of test cases: 10
Time limit: 1 second

In Western chess, a knight can move from position (x, y) to (u, v) if $(x - u) \times (y - v) = \pm 2$ and $x, y, u, v \in \{0, 1, 2, \cdots, 7\}$. For example, a knight at $(4, 5)$ can move to one of the positions $(2, 4), (2, 6), (3, 3), (3, 7), (5, 3), (5, 7), (6, 4),$ *and* $(6, 6)$. A pony knight is a special piece, and a pony knight at position (x, y) can only move to $(u, v) = (x - 1, y - 2)$ or $(u, v) = (x - 2, y - 1)$ if $u, v \geq 0$. For example, a pony knight at position $(4, 5)$ can only move to $(3, 3)$ or $(2, 4)$. A pony knight at position $(2, 1)$ can only move to $(0, 0)$. A pony knight at position $(0, 3)$ cannot move anymore.



(x, y)	(u, v)
$(4, 5)$	$(3, 3)$ or $(2, 4)$
$(3, 3)$	$(1, 2)$ or $(2, 1)$
$(2, 4)$	$(1, 2)$ or $(0, 3)$
$(1, 2)$	only $(0, 0)$
$(2, 1)$	only $(0, 0)$
$(0, 3)$	none

Now Alice and Bob play a game on a large chess board $n \times n$, which positions are labeled as (x, y) where $0 \leq x, y < n$. They randomly place a pony knight at position (x, y). Alice and Bob move the pony knight alternatively, but Alice first. The one who can not move the pony knight anymore loses the game. Assume that Alice and Bob are both very smart and play optimally.

Take $(x, y) = (3, 3)$ as an example. Alice can move the pony knight to $(1, 2)$ or $(2, 1)$. Bob will move the pony knight to $(0, 0)$. Alice cannot move anymore and loses the game.

Take $(x, y) = (2, 4)$ as an example. Possible target positions are $(1, 2)$ or $(0, 3)$. But smart Alice will move the pony knight to $(0, 3)$. Bob cannot move anymore and loses the game.

Take $(x, y) = (4, 5)$ as an example. Possible target positions are $(3, 3)$ or $(2, 4)$. Smart Alice will move the pony knight to $(3, 3)$ since Bob will absolutely lose the game at $(3, 3)$ according to the above examples.

Now, at position (x, y), Alice wants to know whether she has a strategy to absolutely win the game. If Alice can win the game absolutely, then how many ways can Alice win the game absolutely?

Take $(x, y) = (4, 5)$ as an example. There are two ways such that Alice can win the game absolutely. $(4, 5) \to (3, 3) \to (1, 2) \to (0, 0)$ and $(4, 5) \to (3, 3) \to (2, 1) \to (0, 0)$. Although Alice can win the game along with the way $(4, 5) \to (2, 4) \to (1, 2) \to (0, 0)$, smart Bob won't let it occur.

The following table may be helpful to solve this problem.

10	**1**	**1**	**1**	**2**	**2**	**2**	**4**	**10**	**4**	**27**	**48**
9	**1**	**1**	**1**	**2**	**2**	**2**	**4**	**9**	**24**	**8**	**27**
8	**1**	**1**	**1**	**2**	**2**	**2**	**10**	**4**	**18**	**24**	**4**
7	**1**	**1**	**1**	**2**	**3**	**2**	**9**	**14**	**4**	**9**	**10**
6	**1**	**1**	**1**	**2**	**3**	**7**	**4**	**9**	**10**	**4**	**4**
5	**1**	**1**	**1**	**3**	**2**	**6**	**7**	**2**	**2**	**2**	**2**
4	**1**	**1**	**1**	**3**	**4**	**2**	**3**	**3**	**2**	**2**	**2**
3	**1**	**1**	**2**	**2**	**3**	**3**	**2**	**2**	**2**	**2**	**2**
2	**1**	**1**	**2**	**2**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
	0	1	2	3	4	5	6	7	8	9	10

Input File Format

The first line has an integer denoting the number of test cases to follow. For each test case, there are 3 integers, x, y, and d, where $0 \leq x, y \leq 2^{202}$, and $0 < d \leq 20221015$.

Output Format

For each test case, a pony knight is placed at the position (x, y). If Alice has a strategy to win the game absolutely, please output "W" and the number of ways modulo d. If Alice has no strategy to win the game absolutely, please output "L".

Sample Input

```
9
0 0 9999
0 3 9999
1 2 9999
2 1 9999
2 4 9999
3 3 9999
4 5 9999
7 8 9999
7 8 3
```

Output for the Sample Input

```
L
L
W 1
W 1
W 1
L
W 2
W 4
W 1
```

Problem K
Subsequence
Max no. of test cases: 14
Time limit: 2 seconds

You are given three integer sequences A, B and C, where an integer can appear at most once in the same sequence. Your job is to find a common subsequence of A, B and C whose length is maximized. Specifically, let $A = (a_1, a_2, \ldots, a_r)$, $B = (b_1, b_2, \ldots, b_s)$ and $C = (c_1, c_2, \ldots, c_t)$ for some $r, s, t \geq 1$. A common subsequence $D = (d_1, d_2, \ldots, d_\ell)$ of A, B and C is one such that you can find indexes $i_1 < \cdots < i_\ell$, $j_1 < \cdots < j_\ell$ and $k_1 < \cdots < k_\ell$ with $a_{i_t} = b_{j_t} = c_{k_t} = d_t$ for $1 \leq t \leq \ell$. (Note that it is possible that $\ell = 0$, which indicates the empty subsequence.) *A longest common subsequence* is one where ℓ is maximized. If there are more than one candidate, please output the lexicographic smallest one. For example, let $A = (3, 4, 2)$, $B = (4, 3, 2)$ and $C = (3, 4, 2)$. Then $(3, 2)$ is a common subsequence. Similarly, $(4, 2)$ is also a common subsequence. Notice that it is impossible to find a longer one of length 3. Hence, *both* of them are longest common subsequences. You should output $(3, 2)$ in this example because it is the lexicographic smallest one. (Note that lexicographic order is also called the dictionary order.)

Input File Format

The first line has an integer denoting the number of test cases to follow. For each test case, its first line specifies r, s, t ($1 \leq r, s, t \leq 5 \times 10^4$), which are the lengths of sequences A, B and C, respectively. Its second to fourth lines list sequences A, B and C, respectively. All numbers in the same line are space-delimited, and numbers in sequences are all 32-bit signed integers.

Output Format

For each test case, output the longest common subsequence of A, B and C which is lexicographically smallest in one single line. If the output is an empty sequence, then simply output a blank line.

Sample Input

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

Output for the Sample Input

```
7 2
2 -1 4
```

Problem L
River
Max no. of test cases: 20
Time limit: 1 second

We want to cross a river. There are n *stops* in the route to cross the river, and the i-th stop has a *cost* c_i. Every stop is either a *pillar* or a *span*. We first determine where the pillars are and the rest are all spans. The distance between two pillars, or a pillar to the bank of the river, is the sum of c_i of those spans between them. Then we buy a board long enough to cover these consecutive spans. Now we can go from the bank of the river and go through all pillars by moving this board, and finally we can cross the river. The cost to build a pillar at the i-th stop is exactly c_i. The cost of the board is exactly its length. Please find a way to cross the river at the *minimum* cost.

Let us consider an example. Let there be six stops with costs 5, 10, 20, 5, 15, and 5. Now we decide to build two pillars at the third and the sixth stops, so the pillar cost is $20 + 5 = 25$. Since the sum of costs of the first and second stops is $5 + 10 = 15$, but the sum of costs of the fourth and fifth stops is $5 + 15 = 20$, we need a board of length at least 20 to cross the river. Finally, the total cost is the pillar cost and the board cost, i.e., $25 + 20 = 45$.

Input File Format

The first line has an integer denoting the number of test cases to follow. For each test case, there are two lines of input. The first line has the number of stops n ($n \leq 500$) and the second line has n integers ($0 \leq n \leq 100$) denoting the costs.

Output Format

For each test case, output the minimum cost to cross the river in a line.

Sample Input

```
1
6
5 10 20 5 15 5
```

Output for the Sample Input

```
35
```

Problem M
One-Day Trips
Max no. of test cases: 15
Time limit: 13 seconds

Γ is a city that consists of N landmarks. We say that landmarks x and y are *nearby* if some road connects x and y without passing through other landmarks. There is a souvenir store in each landmark. In each store, there are a number of different kinds of souvenirs on sale. In Γ, surprisingly, for each kind of souvenir S, there are at most U pairs of nearby landmarks whose souvenir stores disagree on S as their on-sale souvenirs. That is, one store in the pair has S on sale but the other does not.

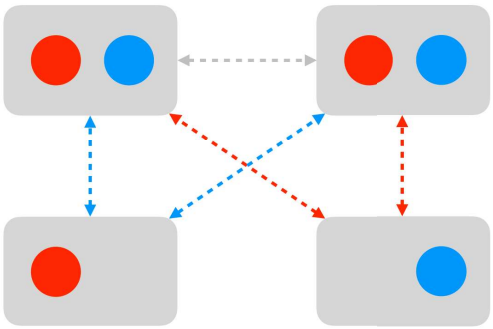


Figure 1: This figure illustrates an example of $U = 2$. The number of pairs of nearby landmarks (depicted as a pair of grey rectangles connected with an edge) whose souvenir stores disagree on the red souvenir (depicted as red disks) as their on-sale souvenirs is 2, and that of blue souvenir also is 2.

John is a tour guide, and he plans many one-day trips, all of which satisfy that:

1. there are exactly three landmarks in the trip, and
2. every two of the three landmarks are nearby.

Surprisingly, in Γ, if a trip satisfies the above two conditions, then it satisfies also that there exists a sourvier S so that at least one of the three landmarks has S on sale and at least one of the three landmarks does not have S on sale. John wonders how many one-day trips that satisfy John's conditions can be found in Γ. It may be worth noting that we do not have the information of the on-sale souvenirs in each store on hand.

Input File Format

The first line contains an integer denoting the number of test cases to follow. For each test case, there are two lines of input. In the first line, there are three positive integers N, M, and U where

- $3 \leq N \leq 10^4$ denotes the number of landmarks,
- $0 \leq M \leq \binom{N}{2}$ denotes the number of nearby landmarks, and
- $U \leq 16$ denotes an upper bound on the number of pairs of nearby landmarks whose souvenir stores disagree on S as their on-sale souvenirs for each souvenir S.

The landmarks are numbered from 1 to N. In the second line, there are M pairs of integers i and j indicating that the i-th and j-th landmarks are nearby.

Output Format

For each test case, output an integer on a line that counts the number of one-day trips that satisfy John's conditions.

Sample Input

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

Output for the Sample Input

```
2
```

Problem N
Cluster PK Cluster
Max no. of test cases: 20
Time limit: 1 second

In social network research, a *cluster* (*community*) is usually represented by a *complete graph* such that the members in this cluster are represnted by vertices, and relationships between members are represented by edges. The two clusters G_1 and G_2 may have some common members that belong to both G_1 and G_2.

For futher developing a software to analyze the relationship among clusters, Dr. Ray, a big-data analyst in the Nonprofit-Cluster-Pk-Cluster (NCPC) company defines some operations on clusters. For two clusters $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, the *union of* G_1 *and* G_2, denoted by $G_1 \cup G_2$, is the graph with vertex set $V_1 \cup V_2$ and edge set $E_1 \cup E_2$; and the *intersection of* G_1 *and* G_2, denoted by $G_1 \cap G_2$, is the graph with vertex set $V_1 \cap V_2$ and edge set $E_1 \cap E_2$.

Note that if $G_1 \cap G_2$ is empty, then G_1 and G_2 are two distinct components. Otherwise, if $G_1 \cap G_2$ is non-empty, then one may contain the other, or both have a non-empty and proper intersection. For ease of analysis, Ray would like to assign lables to members of clusters. A labeling of G is said to be *proper* when we assign labels to the vertices of G so that if u and v are adjacent, then the labels assigned to u and v are different.

Note that the vertices in a cluster $G = (V, E)$ are distinguished by members. Consequently, two proper labelings of a cluster will be considered different in the following sense: a proper labling (of the vertices of G) that uses at most λ labels is a function f, with domain V and codomain $\{1, 2, 3, \ldots, \lambda\}$, where $f(u) \neq f(v)$, for adjacent vertices $u, v \in V$. Proper labelings are then different in the same way that these functions are different. The number of different ways to label a graph using λ labels is called the *critical number* of the given graph. For example, if $G = (V, E)$ with $|V| = n$ and $E = \emptyset$, then G consists of n isolated vertices, and by the rule of product, the critical number of G equals λ^n.

Given (1) $|V_1|$ and $|V_2|$, (2) the number $|V_1 \cap V_2|$, and (3) the number of available labels λ, your task is to provide a computer program to compute the number of different ways to label $G_1 \cup G_2$. Note that if the result is larger than or equal to $10^9 + 7$, you should output the value modulo $10^9 + 7$, that is, the remainder obtained using the actual value divided by $10^9 + 7$.

Technical Specification

1. A complete graph is a graph is a simple graph whose vertices are pairwise adjacent.
2. $1 \leq |V_i| \leq 10000$ for $i = 1, 2$.
3. $0 \leq |V_1 \cap V_2| \leq \min\{|V_1|, |V_2|\}$.
4. $0 \leq \lambda \leq 10^6$.

Input Format

The first line contains an integer n, which represents the number of test cases. Each test case contains a single line with four non-negative integers $|V_1|$, $|V_2|$, $|V_1 \cap V_2|$, and the number of available labels λ. Any two consecutive integers are separated by exactly one space.

Output Format

For each test case, output the corresponding critical number. Note that if the result is larger than or equal to $10^9 + 7$, you should output the value modulo $10^9 + 7$, that is, the remainder obtained using the actual value divided by $10^9 + 7$.

Sample Input

```
1
2 2 0 4
```

Sample Output for the Sample Input

```
144
```