Problem A. Simplified Genome Translation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

The translation is a critical step for transferring the genome information (mRNA) into a biology unit (protein). Specifically, it parses three RNA nucleotides into one amino acid according to the codon table (Table 1).

amino acid	RNA nucleotides
F	UUU UUC
L	UUA UUG CUU CUC CUA CUG
I	AUU AUC AUA
M	AUG
V	GUU GUC GUA GUG
S	UCU UCC UCA UCG AGU AGC
P	CCU CCC CCA CCG
T	ACU ACC ACA ACG
A	GCU GCC GCA GCG
Y	UAU UAC
H	CAU CAC
Q	CAA CAG
N	AAU AAC
K	AAA AAG
D	GAU GAC
E	GAA GAG
C	UGU UGC
W	UGG
R	CGU CGC CGA CGG AGA AGG
G	GGU GGC GGA GGG
STOP	UAA UAG UGA

Table 1: The codon table.

For example, given an RNA sequence, R = `CUCAGCGUUACCUAGUUUCAUUGUGCU`, its three code parsing is `CUC` `AGC` `GUU` `ACC` `UAG` `UUU` `CAU` `UGU` `GCU`, and its translated amino acid is P = `LSVT`. Notice `UAG` is a stop codon that stops the translation process. There are three stop codons, `UAA`, `UAG`, and `UGA`, in Table 1.

Input

The first line contains an integer T, which represents the number of test cases. Each test case below has one line, an RNA sequence, R.

Constraints

- $1 \leq T \leq 50$.
- R is the sequence of the alphabet $\Sigma = \{$`A, C, G, U`$\}$.
- $|R| = 3 * n$, where $1 \leq n \leq 333$.
- P is the translated amino acids from the R terminated by STOP codons if existing (i.e., the first test case) or complete translation if no STOP codon. STOP codon will not be at the beginning of P.

- All string characters are uppercase letters.

Output

Each test case outputs the corresponding translated amino acid, P.

Examples

standard input	standard output
5 UUUUAACACUUUAUCACUUAACACCAC CAAAAUAUGAAAAAU AUGUACUUUGCGUUUCACUAA UUGCACUACUAC UACGUGGGUAUC	F QNMKN MYFAFH LHYY YVGI

Problem B. Multi-Ladders

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	1024 megabytes

In order to attract customers in a store, the owner decided to order a special neon sign to be placed in front of the store. This special neon sign is composed by several ladder-type components. Each ladder-type component can be represented by a ladder graph L_n that is a planar, undirected graph with $2n$ vertices and $3n-2$ edges. Each vertex represents a light bulb of the neon sign, and each edge represents a wire connecting two light bulbs. The ladder graph can be obtained as the Cartesian product of two path graphs, one of which has only one edge: $L_n = P_n \times P_2$, where P_n is a path of n vertices. Figure 1 shows L_6. The main frame for holding k ladder-type components is k-regular polygon. Each edge of k-regular

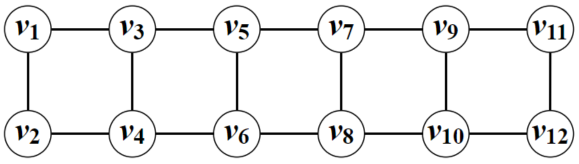


Figure 1: L_6.

polygon is combined with the top edge of L_n. Figure 2 shows the case with $k = 3$ and $n = 4$, and Figure 3 shows the case with $k = 4$ and $n = 3$. For ease of description, the neon sign is represented by a graph $G = (V, E)$ comprised by the bulbs (vertices) and wires (edges). To make the neon lights more dazzling,

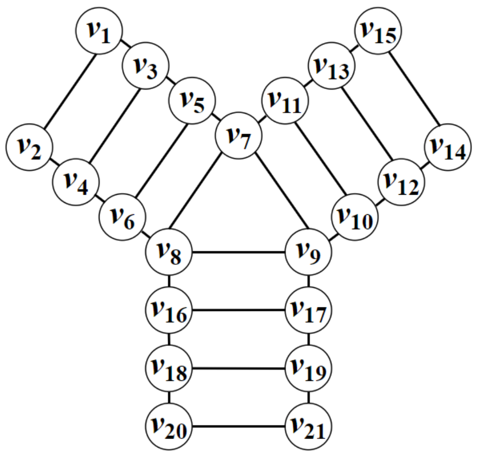


Figure 2: An example of $k = 3$ and $n = 4$.

the designer, Ray, came up with a way to specify the color of the bulbs (vertices) in G. Ray would like to assign colors to blubs such that the following condition hold. A coloring of G is said to be *proper* if we assign colors to the vertices of G so that if u and v are adjacent, then the colors assigned to u and v are different. The bulbs in $G = (V, E)$ are distinguished, that is, the bulbs are named by labels $v_1, v_2, \ldots v_m$. Consequently, two color assignments of bulbs in G will be considered different if a proper coloring of the bulbs of G that uses at most λ colors 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 colorings are then different in the same way that these functions are different. The maximum number of different ways to color G using λ colors is called the *critical number* of G.

Given (1) n (for L_n), (2) k (for k-regular polygon), and (3) the number of available colors λ, your task is to compute the critical number of G. Note that if the result is larger than or equal to $10^9 + 7$, you

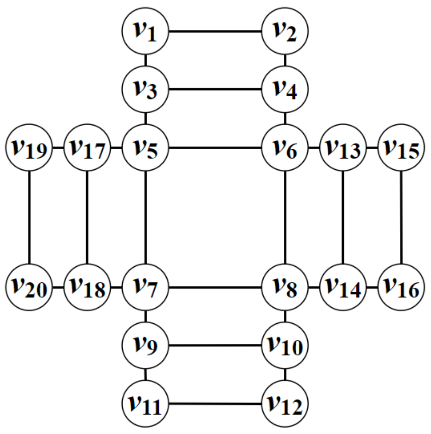


Figure 3: An example of $k = 4$ and $n = 3$.

should output the value modulo $10^9 + 7$, that is, the remainder obtained using the actual value divided by $10^9 + 7$.

Input

The first line of the input file contains an integer L ($L \leq 20$) that indicates the number of test cases as follows. For each test case, the first line contains three integers (separated by whitespaces) representing n, k, and λ.

Constraints

- $1 \leq n \leq 1000000000$.
- $3 \leq k \leq 1000000000$ for each test case.
- $0 \leq \lambda \leq 1000000000$.

Output

The output contains one line for each test case. Each line contains one non-negative integer representing the critical number of G. 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 $10^9 + 7$.

Examples

standard input	standard output
1 2 3 3	162

Problem C. Distance Calculator

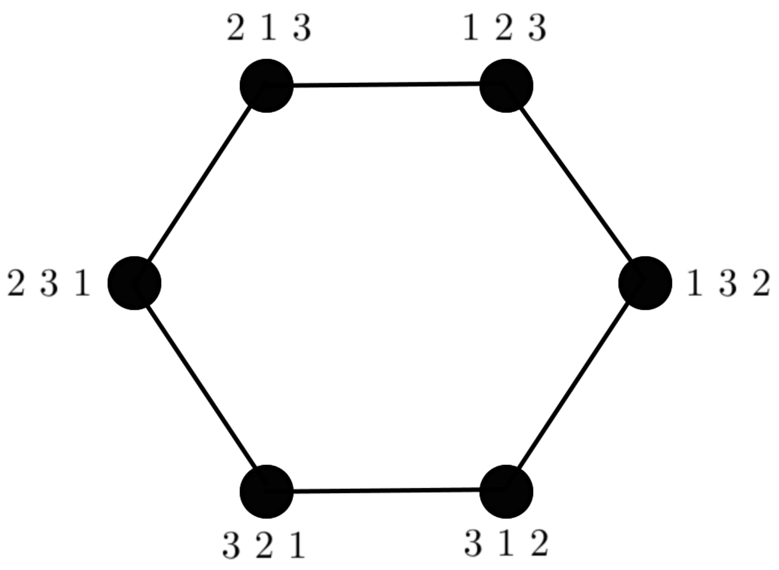
Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Wonder kingdom has $n!$ cities. Each city is encoded with n number $d_1 d_2 \dots d_n$ which is a permutation of $1\ 2 \dots n$. The castle of Wonder kingdom is located in the city encoded as $1\ 2 \dots n$. Let $a_1 a_2 \dots a_n$ and $b_1 b_2 \dots b_n$ be the codes of cities A and B, respectively. A road with distance one is built between cities A and B in the kingdom if and only if there exists an i, $1 \le i < n$, such that the following two conditions are satisfied.

1. $a_i = b_{i+1}$ and $b_i = a_{i+1}$;
2. $a_j = b_j$ for $j \in \{1, 2, \dots, n\} \setminus \{i, i+1\}$.

One day the king invites all mayors for a meeting in the castle. Please help mayors calculate their travel distance to the castle. Notice that the city of the castle is encoded with $1\ 2\ 3 \dots n$.

See the following example. There are 6 cities in the kingdom. Each city is encoded with a permutation of 1 2 3.



Input

The first line contains an integer m which represents the number of test cases. Each test case below contains two lines. For each test case, the first line is an integer n, $3 \le n \le 100$ which indicates that there are $n!$ cities in the kingdom and the second line consists of different n numbers in $\{1, 2, \dots, n\}$ with a space between two numbers which indicates the encoding of the given city.

Constraints

- $1 \le m \le 50$.
- $1 \le n \le 100$.

Output

For each test case, output one line containing an integer which indicates the distance between the given city and the castle.

Examples

standard input	standard output

5
3
3 1 2
4
4 3 2 1
5
4 1 2 3 5
7
2 6 1 5 4 3 7
10
3 2 1 5 7 6 4 10 8 9
	2
	6
	3
	8
	9

Problem D. Tangle: A DAG for storing transactions

Input file:	standard input
Output file:	standard output
Time limit:	3 seconds
Memory limit:	1024 megabytes

In general, a tangle-based cryptocurrency works in the following way. There is a directed acyclic graph (DAG) that we call the Tangle. The network of DAG is composed of nodes, that is, nodes are entities that issue and validate transactions. The transactions issued by nodes constitute the site set of the Tangle graph, which is the ledger for storing transactions. The edge set of the Tangle is obtained in the following way: when a new transaction X arrives, it must approve two previous transactions. These approvals are represented by directed edges, as shown in Figure 4(a). If there is not a directed edge between transaction A and transaction G, but there is a directed path of length at least two from A to G, we say that A indirectly approves G. In Figure 4(a), A direct approves B and D and indirectly approves F, G, H, and I. Let us define "Tips" as unapproved transactions in the Tangle graph. In the *Tangle* snapshot of Figure 4(a), the Tips are A and C. When the new transaction X arrives and approves A and C in the Tangle snapshot of Figure 4(b), X becomes the only Tip.

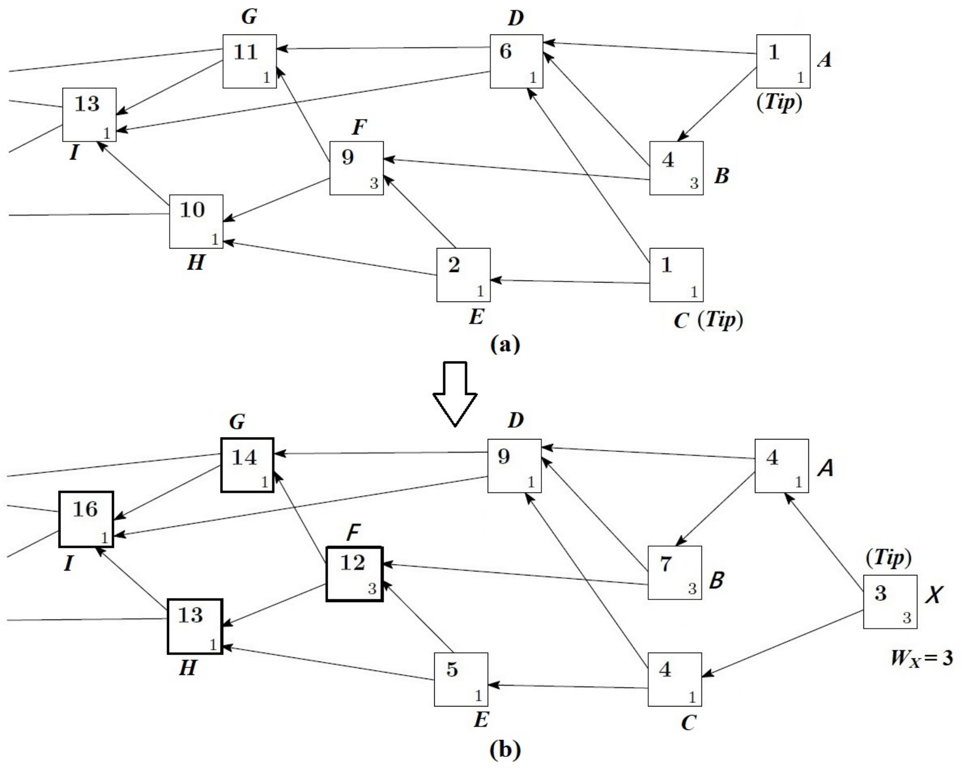


Figure 4: A *Tangle* graph with weight assignments before and after a newly issued transaction X. The boxes represent transactions, the small number in the South-East(SE) corner of each box denotes own weight, and the bold number denotes the cumulative weight.

We define the weight of a transaction, and related concepts. The weight of a transaction is proportional to the amount of work that the issuing node invested into it. It is only important that every transaction has a positive integer, its weight, attached to it. In general, the idea is that a transaction with a larger weight is more "important" than a transaction with a smaller weight. Assume that the transactions X has an own weight equal to W_X, where $1 \le W_X \le W_{max}$. The cumulative weight CW_X of X is the sum of its weight and weights of all transactions which directly or indirectly approves it. The calculation of cumulative weight is illustrated in Figure 4. The boxes represent transactions, the small number in the South-East corner of each box denotes own weight, and the bold number denotes the cumulative weight. For example, transaction F is directly or indirectly approved by transactions A, B, C, and E.

The cumulative weight of F is $CW_F = 3+1+3+1+1 = 9$, which is the sum of the own weight of F and the weights of A, B, C, and E, respectively. After the new transaction X ($W_X = 3$) approved, the cumulative weight of all other transactions increases by 3. The cumulative weight of a transaction is a measure of the computational effort behind it. So, once it reaches a threshold value TH, the transaction is marked as confirmed as the probability of malicious modification is low. In other words, as a transaction receives additional approvals, it is accepted by the system with a higher level of confidence. Approved transactions which have cumulative weight less than the upper limit are called unconfirmed transactions. As shown in Figure 4, transactions F, G, H, and I are confirmed, if we assume $TH = 12$. In the beginning of the $Tangle$, there was an address with a balance that contained all of the tokens. The Genesis transaction sent these tokens to several other "founder" addresses. All of the tokens were created in the Genesis transaction. Therefore, the "Genesis" transaction is approved either directly or indirectly by all other transactions (in Figure 5). Initially when a node creates a transaction, it selects two Tips (unapproved transactions) based on a Tip-selection algorithm. Each new transaction should approve two previous transactions, which are represented by directed edges. Other nodes in the network will verify the transaction and add it as a new Tip in the DAG. When a transaction is added as a Tip in the DAG, it is unapproved. When any further transactions select that Tip, it becomes approved. The cumulative weight associated with a transaction will be updated whenever a transaction approves it directly/indirectly.

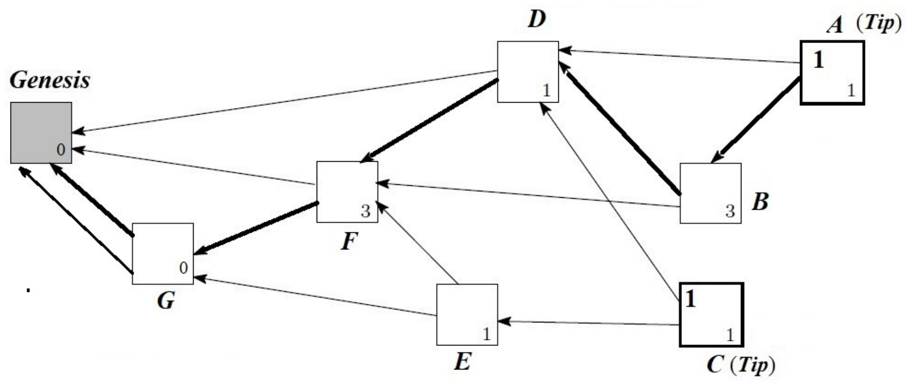


Figure 5: A Tangle graph with weight assignments to each transaction

For convenience, the initial Tangle is shown in Figure 6(a). A new created transaction X will select two Tips Y and Z (unapproved transactions) based on a Tip-selection algorithm. In Figure 6 (b) shows the notations of weights and cumulative weights for X, Y, and Z, respectively.

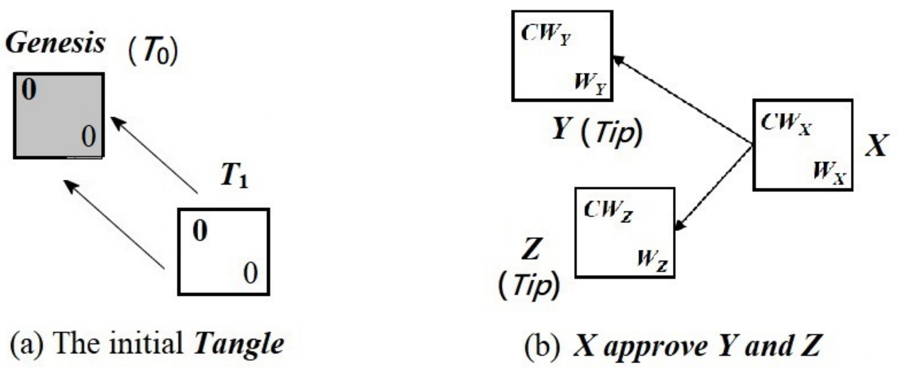


Figure 6: A Tangle graph with weight assignments to each transaction

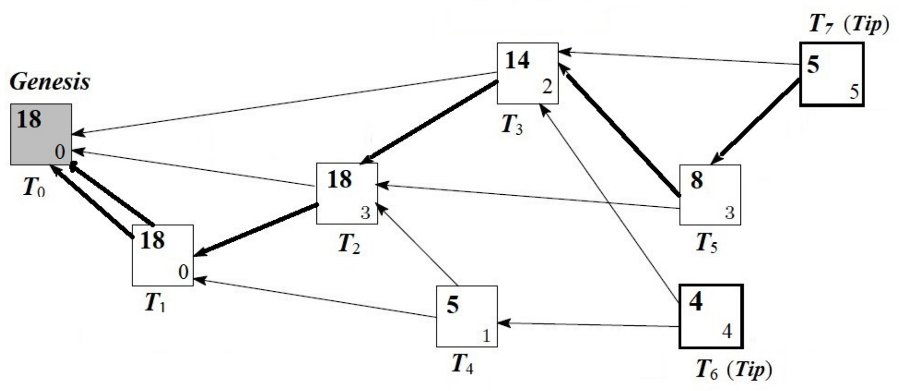


Figure 7: The Tangle graph for Sample Input 1, where $TH = 12$ ($CW_X \geq TH$ for T_2 and T_3).

Input

The first line of the input is a pair of integer n TH, where n denotes the number of transactions and TH denotes the threshold value of cumulative weight. The transactions are input in increasing order of X. Each transaction is in one line, which contains a list of integers: X Y Z W_X, where $2 \leq X \leq 10000$, $0 \leq Y, Z \leq X - 1$, $Y \neq Z$, and $1 \leq W_X \leq 5$.

Constraints

- $1 \leq n \leq 10000$
- $2 \leq X \leq n + 1$
- $0 \leq Y, Z \leq X - 1$, $Y \neq Z$

Output

After processing the n transactions, for a transaction T_X in $\{T_2, T_3 \ldots, T_n\}$ if $CW_X \geq TH$, then output the list of values: X CW_X in increasing order of X. Each line contains one transaction. Finally, output the total number of confirmed transactions.

Examples

standard input	standard output

6 12
2 0 1 3
3 0 2 2
4 1 2 1
5 2 3 3
6 3 4 4
7 3 5 5
	2 18
	3 14
	2

Problem E. Printing Stickers

Input file:	`standard input`
Output file:	`standard output`
Time limit:	30 seconds
Memory limit:	1024 megabytes

You work in the International Collaborative Printing Company (ICPC) which prints documents, booklets, and stickers for all kinds of campaigns. Recently, your company acquired a new sticker printing machine. This printing machine prints pre-configured triangle shades onto a huge sticker sheet. The sheet can be described as a grid of $M \times N$ cells, where M is the number of rows and N is the number of columns. Each cell is divided into two triangles in some specified way, and the machine fills some of the triangles with inks. See Figure 8 as an example.

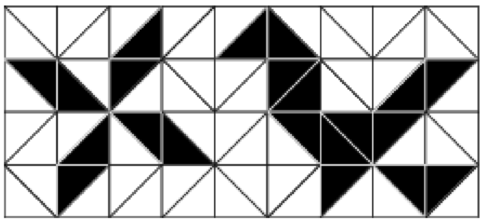


Figure 8: A printed sticker sheet before cutting into pieces of individual stickers.

Since each sticker is relatively small, the printer prints multiple stickers on a sheet, and then cuts them off from the sheet. The cut can be made along the boundary of the triangles. However, to ensure the high quality of a sticker, it is required that all cuts should not be cut along the edge of any filled triangle. Notice that after cutting the sticker sheet into lots of smaller pieces, the pieces without any filled triangles are thrown away. Each remaining sticker has at least one filled triangle. Moreover, on each sticker, all the filled triangles are connected. That is, for any two filled triangles there is a sequence of filled triangles on the same sticker such that every pair of neighboring triangles share at least one common point. Figure 9 demonstrates valid and invalid ways to cut the sticker sheet.

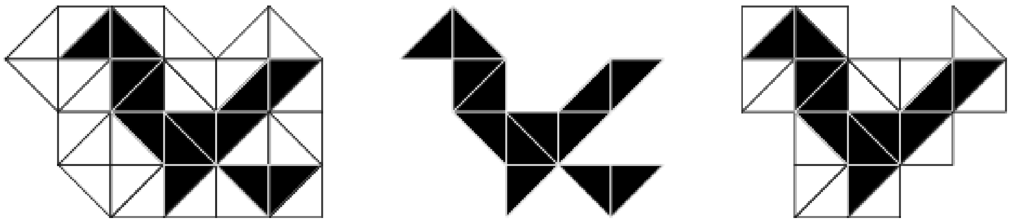


Figure 9: The leftmost sticker is valid, but the other two stickers are invalid.

After cutting off all the stickers from the sheet, an additional polishing step is applied to the boundary of each sticker. Notice that it is possible for a sticker to have holes (see Sample Input 2). In this case, the boundary of the holes also needs to be polished. Cutting the sheet is free, but the polishing step is associated with some costs. Specifically, the boundary of each sticker can be described by a collection of segments. Each segment refers to a horizontal cut, a vertical cut, or a diagonal cut. Each horizontal cut is associated with a polishing cost of H and each vertical cut is associated with a polishing cost of V. The diagonal cuts have slightly different polishing costs $\{D_{ij}\}$, depending on the location of the cut. In particular, the diagonal cut at the ith row and the jth column has a polishing cost D_{ij}. The cost of polishing a sticker is then defined by the sum of all polishing costs associated with the segments. See Figure 10 and Figure 11.

Please write a program that helps the company compute for each sticker, the minimum cost of cutting and then polishing. One can prove that there exists an optimal way to cut off all the stickers from the sticker sheet at once, such that every sticker achieves the minimum polishing cost simultaneously.

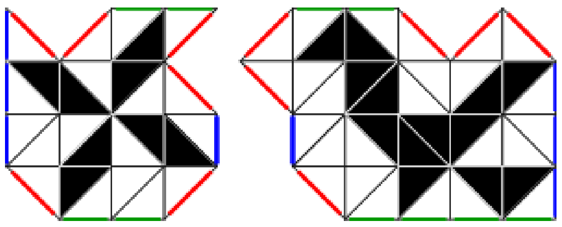


Figure 10: An illustration of computing the cost for polishing stickers. Suppose $H = V = 10$, and each diagonal cut has cost $D_{ij} = 1$. The first sticker on the left is comprised of 4 horizontal segments, 4 vertical segments, and 6 diagonal segments, so the total cost is 86. The second sticker on the right has a cost of 106.

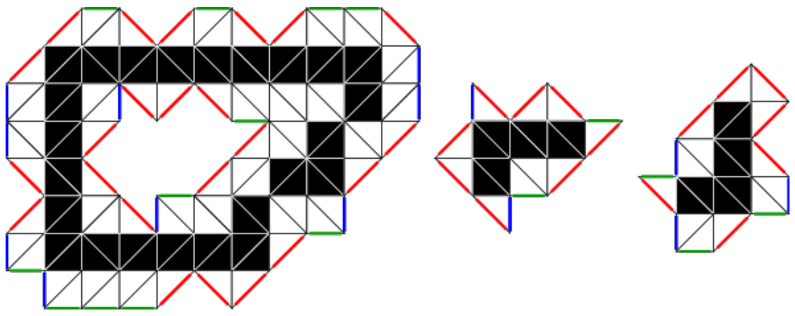


Figure 11: An illustration to Sample Input 2. Notice that the boundary of holes requires polishing. The polishing costs for each sticker are 723, 196, and 214 respectively.

Input

The first line contains an integer T, which represents the number of test cases. For each test case, the first line contains four integers M, N, H, and V, which represent the number of rows, the number of columns, the polishing cost per horizontal segment, and the polishing cost per vertical segment respectively. Then each of the following M lines contains a string of length N, where each character in the string is either '/' or '\'. The jth character of the ith string denotes the direction of the diagonal. Then another M lines follow. In each of the M lines, there is a string of length $2N$. For each cell location (i, j) in the grid, the $(2j + 1)$th character of the ith string denotes whether or not the left-hand-side triangle of the diagonal is filled or not, whereas the $(2j + 2)$th character of the ith string denotes whether or not the right-hand-side triangle of the diagonal is filled. If a triangle is filled with ink then the corresponding character is '#', otherwise, the corresponding character is '.'. Finally, there are M lines. In the ith line there are N integers $D_{i1}, D_{i2}, \ldots, D_{iN}$, denoting the cost of polishing each diagonal.

Constraints

- $1 \leq T \leq 50$.
- $M \geq 2$; $N \geq 2$; $4 \leq M \times N \leq 10{,}000$.
- $1 \leq H, V, D_{ij} \leq 1{,}000$ for all $1 \leq i \leq M$ and $1 \leq j \leq N$.
- For each test case, at least 1 and at most 1,000 stickers will be produced from the machine.
- There is no filled triangle whose edge is the boundary of the entire sticker sheet.

Output

Each test case outputs two lines. In the first line, there is an integer k, denoting the total number of stickers. In the second line, there are k integers $c_1, c_2, \ldots, c_k$, denoting the minimum cost for polishing all stickers, sorted in the non-decreasing order.

Examples

standard input	standard output
1 4 9 10 10 \////\\/\ \\/\//\// //\\/\\/\ \///\//\/ .....#...##....... .##.#.....##...##. ...#.##....####... ..#.........#..##. 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	2 86 106
1 8 12 20 19 \//\//\/\\\/ //\/\\\///// ///\/\\\\//\ \//\\\/\/\/\ \\\\\///\/// ///\\\\\\\\\ //\/////\\\\ /////\////// ....##################.... ..##..............##.... ..##..######....##...... ..##..##......####..##.. ..##........##......##.. ..############....####.. .... 11 12 13 15 14 12 17 16 14 13 11 10 12 13 15 14 13 17 18 17 15 14 12 16 16 17 18 17 15 14 13 11 16 17 18 19 19 11 12 13 15 14 16 16 17 18 14 13 13 20 16 15 14 14 13 11 10 12 12 13 16 17 17 14 15 16 19 12 14 11 14 16 18 17 14 14 16 19 18 14 15 13 12 14 15 16 17 15 11 18 19 16 16 14 14 20	3 196 214 723

Problem F. AA Country and King Dreamoon

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

There is a prosperous country, AA, which is consisting of n cities(numbering from 1 to n) and $n - 1$ roads. Because of good road design, you can start from any city and go through several roads to reach the destination you want to reach.

The king of the AA country, Dreamoon, wanted to inspect the law and order of all the cities tomorrow, so he made a route plan, which consists of a path with $2 \times n - 2$ edges, starting and ending at city 1(the capital city of AA country). For the convenience of recording, Dreamoon writes the numbers of the cities on this path on paper in order.

However, Dreamoon spilled the milk tea on the paper unfortunately, there are consecutive numbers that are not clear now. Thus, he asks you to help restore these unclear numbers in order to have a good journey.

If there are multiple solutions, please output the solution in the smallest lexicographical order. In addition, it is guaranteed to have at least one solution for the given input.

Input

The first line contains one integer T — the number of test cases. The following is a description of the input data sets.

Each test case contains two lines. The first line of each test case contains one integer n — the number of cities in AA country. The second line of each test case contains $2 \times n - 1$ integers between 0 and n - the i-th number representing the i-th city in this path. If the i-th number is 0, the number cannot be seen clearly because of the overturned milk tea. The input guarantees that all 0s are consecutive and there is at least one 0.

Constraints

- $1 \le T \le 30,000$.
- $1 \le n \le 300,000$.
- The cities are numbered from 1 to n.
- At least one number of city Dreamoon writes on paper is not clear and all unclear numbers are consecutive.
- The sum of n across all test cases does not exceed $300,000$.

Output

For each test case, please output a line containing $2 \times n - 1$ numbers between 1 and n, representing the path you restore. Remember, output the solution in the smallest lexicographical order if there are multiple ones. It is guaranteed to have at least one solution for the given input.

Examples

standard input	standard output

```
9
5
1 2 3 2 0 2 1 5 1
5
1 2 3 0 0 2 1 5 1
5
1 2 0 0 0 2 1 5 1
5
1 2 0 0 0 0 1 5 1
5
1 0 0 0 0 0 1 5 1
5
1 0 0 0 0 0 0 5 1
5
1 0 0 0 0 0 0 0 1
5
1 0 0 0 0 0 0 0 0
5
0 0 0 0 0 0 0 0 0
```
	```
	1 2 3 2 4 2 1 5 1
	1 2 3 2 4 2 1 5 1
	1 2 3 2 4 2 1 5 1
	1 2 1 3 1 4 1 5 1
	1 2 1 3 1 4 1 5 1
	1 2 1 3 1 4 1 5 1
	1 2 1 3 1 4 1 5 1
	1 2 1 3 1 4 1 5 1
	1 2 1 3 1 4 1 5 1
	```

Problem G. Repetitive Elements

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

The genome is a sequence with four nucleic acids, `A`, `T`, `C`, and `G`. Repetitive elements inside the genome are patterns of nucleic acids that occur in multiple copies throughout the genome without overlapping. For example, given a toy genome, $S =$ `TATCGATCGAG`, there are a couple of repeat elements where $R =$ `ATCG` is the longest repetitive element appearing in `TATCGATCGAG` and `TATCGATCGAG` instead of `ATCGA`, whose two appearing positions overlap. Your task is to identify the longest repetitive element R inside the given genome sequence, S.

Input

The first line contains an integer T, which represents the number of test cases. Each test case is a genome sequence, S.

Constraints

- $1 \leq T \leq 50$.
- S is the sequence of the alphabet $\Sigma =$ `{A, T, C, G}`.
- $15 \leq |S| \leq 100$.
- If there are multiple longest repetitive elements, please output the one appearing first (i.e., `CTT` instead of `ATG` in the fifth test case).

Output

Each test case outputs R, a string corresponding to the longest repetitive element for the given genome, S.

Examples

standard input	standard output
5	ATCG
TATCGATCGAGTTGT	GCGA
TCCGCGAGCGAGTCTCTCCATT	CATACG
GTTTCATCATACGAGGCCCCATACGCGCTGG	GAT
AGATGGGATCCTTATG	CTT
GCCCTTAGGCATGGGATGTCGTTTCTTG	

Problem H. Meeting Places

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

Welcome to the 2022 ICPC(International Collegiate Programming Contest) Taoyuan Regional Contest!

ICPC is an algorithmic programming competition for university students. It is designed to showcase university students' creativity, teamwork, and ability to write codes, analyze, and solve problems under pressure.

This year, there are N problem setters(numbering from 1 to N) for Taoyuan regional contest.

In order to make the quality of the whole competition better, the problem setters held several meetings during the preparation.

And how to choose the meeting place is a very important thing, because it is necessary to consider the residence of the problem setters.

For convenience, the organizer divided the problem setters into K groups. Each group is composed of consecutively numbered problem setters, and each problem setter will only appear in one group.

For the same group of problem setters, there will be the same meeting place, and the cost of this group is defined as the farthest distance from the members of the group to the meeting place.

Now, you know the location of N problem setters (as a point on a two-dimensional Cartesian coordinate), and K, can you find the minimum sum of costs for these K groups?

The distance here refers to the Euclidean distance, which means that for two points $(x_1, y_1), (x_2, y_2)$, the distance between them is $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

This question may be a bit difficult for you, in addition, you unexpectedly discover a piece of information, and you don't know if it's useful. That is, the residence of each problem setter will satisfy the certain formula. In the other words, the N points are generated by the following formula:

$X_i = Y_{i-1} \times 233811181 + 1(\text{mod}(2^{31} - 1)), \forall i \geq 2.$
$Y_i = X_i \times 233811181 + 1(\text{mod}(2^{31} - 1)), \forall i \geq 1.$

The above stories are purely fictitious, and any coincidences are purely similar.

Input

The first line contains three integers, N, K, X_1, which represent the number of points, the number of intervals, and the random seed.

Constraints

- $1 \leq K \leq N \leq 2000.$
- $1 \leq X_1 \leq 8831$

Output

Output one floating point which represents the minimum value of the sum of the radii of the smallest enclosing circles in the K segments.

Your answer will be considered correct if its absolute or relative error on both coordinates does not exceed 10^{-6}. Formally let your answer be a, jury answer be b. Your answer will be considered correct if $\frac{|a-b|}{max(1,|b|)} \leq 10^{-6}$.

Examples

standard input	standard output
100 23 213	1319350480.8007326126

Problem I. Cell Nuclei Detection

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	1024 megabytes

International Cancer Pathology Company (ICPC) is a company devoted to developing Artificial Intelligent (AI) models to detect *cancer cells* from the pathology images of tumors. In the preprocessing stage, the AI model will detect cell nuclei on a pathology image and use the so-called "*bounding box*" to enclose each cell nuclei, as shown in Figure 12. A bounding box is a rectangle that is represented as a 4-tuple (x_1, y_1, x_2, y_2), where (x_1, y_1) are the coordinates of its top-left corner, and (x_2, y_2) are the coordinates of its bottom-right corner. A bounding box drawn by the AI model is called a *detected bounding box*.

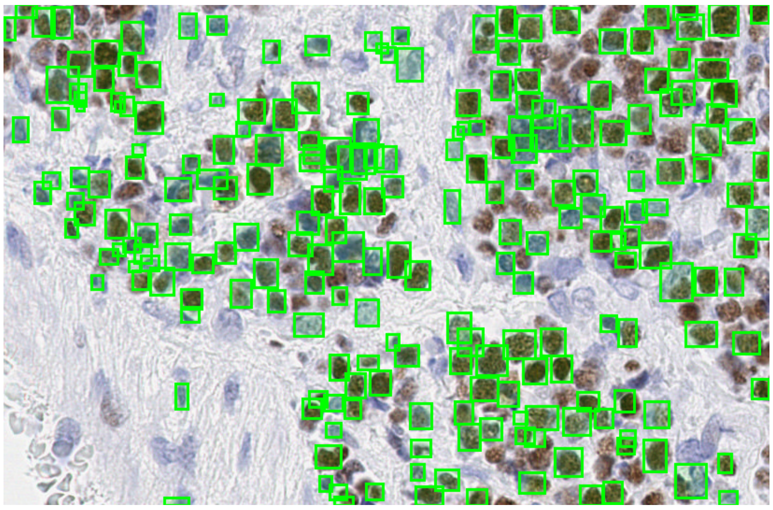


Figure 12: A pathology image with detected bounding boxes.*

To evaluate the performance of the AI model, the researchers of ICPC will prepare a set of test data (pathology images), and medical experts will help to annotate the ground truth (indeed cell nucleus) of the test data by bounding boxes in advance. Those bounding boxes are called *ground-truth bounding boxes*. Then the researchers will apply the AI model to detect cell nuclei on the test data set and see how many ground truth cell nuclei are detected. We say a cell nuclei with ground-truth bounding box G is detected by the AI model if there exists a detected bounding box D overlaps G with at least half the area of G, i.e., $|D \cap G|/|G| \geq 1/2$. Notice that a detected bounding box may overlap with two or more ground-truth bounding boxes, but only one can be said to be detected by this detected bounding box. Figure 13 shows an example of ground-truth bounding boxes and detected bounding boxes. Moreover, the width and height of a bounding box have a limited bound since the size of cell nuclei is limited.

Given a set of ground-truth bounding boxes and a set of detecting bounding boxes, please write a program to determine the maximum number of cell nuclei detected.

Input

The first line contains an integer t, which represents the number of test cases. For each test case, the first line contains a pair of integers m and n, separated by a blank, and represents the numbers of the ground-truth bounding boxes and the detected bounding boxes, respectively. The following $m + n$ lines represent the 4-tuple (x_1, y_1, x_2, y_2) of the bounding boxes' coordinates, in which a blank separates every two consecutive integers. The first m lines represent the ground-truth bounding boxes and the following n lines represent the detected bounding boxes.

* The images in the Figures are cited from the paper of URL: https://www.kitware.com/cell-nuclei-detection-on-whole-slide-histopathology-images-using-histomicstk-and-faster-r-cnn-deep-learning-models/.

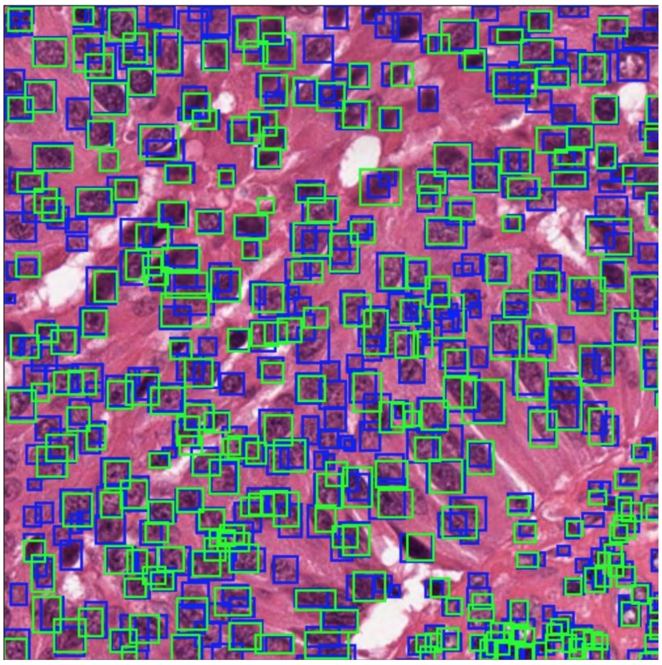


Figure 13: A pathology image with ground-truth bounding boxes (blue boxes) and detected bounding boxes (green boxes).*

Constraints

- $1 \leq t \leq 5$.
- $1 \leq m, n \leq 50000$.
- $0 \leq x_1, y_1, x_2, y_2 \leq 2000$.
- Variables $t, m, n, x_1, y_1, x_2, y_2$ are integers.
- The maximum length of the width and the height of a bounding box is 4.
- No two ground-truth bounding boxes have the containment relation, i.e., one contains the other, and no two detecting bounding boxes.

Output

Each test case outputs one integer k in a line, where k is the maximum number of cell nucleus detected by the given detected bounding boxes.

Examples

standard input	standard output

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

Problem J. Traveling in Jade City

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Jade City has a circular railroad system, which has N stations labeled by $1, 2, \ldots, N$ in the clockwise direction. Due to the city's rapid development in recent years, an extra line that connects Station 1 and Station K $(1 < K \le N)$ is now in use, and apart from these two terminal stations, there are M other stations on the line. These M stations are labeled by $N+1, N+2, \ldots, N+M$, from the direction of Station 1 to Station K. See Figure 14 for an illustration.

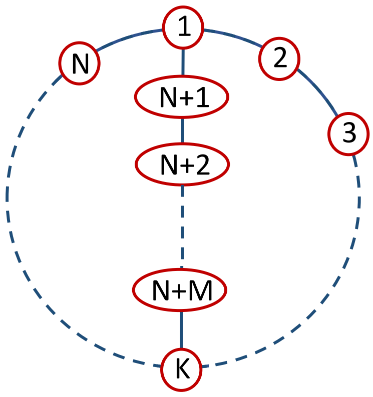


Figure 14: An illustration of Jade City's railroad system

The rails connecting adjacent stations have different lengths, so that traveling along them requires different amount of time. For simplicity, we label these rails as follows:

- c_i: For $1 \le i \le N-1$, the rail between Station i and Station $i+1$;
- c_N: For the rail between Station N and Station 1;
- x_0: For the rail between Station 1 and Station $N+1$;
- x_j: For $1 \le j \le M-1$, the rail between Station $N+j$ and Station $N+j+1$;
- x_M: For the rail between Station $N+M$ and Station K.

Given the traveling time for each of these rails, we want to compute the fastest way to travel from some station u to some station v using this railroad system. However, from time to time, some of the rails may become unavailable due to maintenance, so that the train cannot pass through them. (Initially, all the rails are available.)

Input

The first line contains three integers N, K, and M, as specified in the problem description, followed by an integer Q, representing the total number of query or update operations. The second line contains N non-negative integers, which denote the traveling times t for the rail $c_1, c_2, \ldots, c_N$, respectively. The third line contains $M+1$ non-negative integers, which denote the traveling times t for the rail $x_0, x_1, \ldots, x_M$, respectively. Then, Q lines follow, where each line specifies an operation by the following format:

- If the line begins with the letter "q" and is followed by two integers u and v $(1 \le u, v \le N+M)$: It requests us to output the fastest time to travel from station u to station v, if there is a way. Else, output the word "impossible".

- If the line begins with the letter "c", followed by an integer i ($1 \le i \le N$): It means the state of edge labeled by c_i is toggled. Precisely, if c_i is originally available, then after this operation, c_i becomes unavailable. In contrast, if c_i is originally unavailable, then after this operation, c_i becomes available.
- If the line begins with the letter "x", followed by an integer j ($0 \le j \le M$): It means the state of edge labeled by x_j is toggled. Precisely, if x_j is originally available, then after this operation, x_j becomes unavailable. In contrast, if x_j is originally unavailable, then after this operation, x_j becomes available.

Constraints

- $1 \le N, K, M, Q \le 10^6$;
- The traveling time of any rail is a non-negative integer at most 200.

Output

For each query, output on a separate line the corresponding answer.

Examples

standard input	standard output
4 3 1 9 2 3 8 4 1 1 q 3 4 x 0 q 3 4 c 3 q 3 4 c 1 q 3 4 x 0 q 3 4	6 8 9 impossible 6

Problem K. Group Guests

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	1024 megabytes

There are n guests at a party. Each of the n guests has 2 different hobbies. No two guests have the same set of hobbies. The party host would like to divide the guests into groups of size 2 or 3 so that every two guests in a group have a hobby in common. That is, each group belongs to one of the following cases:

- Type A: a group of two guests who have one hobby in common;
- Type B: a group of three guests, all of whom have a hobby in common;
- Type C: a group of three guests, all pairs of whom have a hobby in common, excluding the case of type B.

The party host would like to reduce the number α of guests who are not assigned to any group to the minimum possible. If multiple solutions have the minimum α, we need to find a solution that minimizes the number β of type-B groups used in the solution while retaining α minimized. Write a program to determine the two numbers.

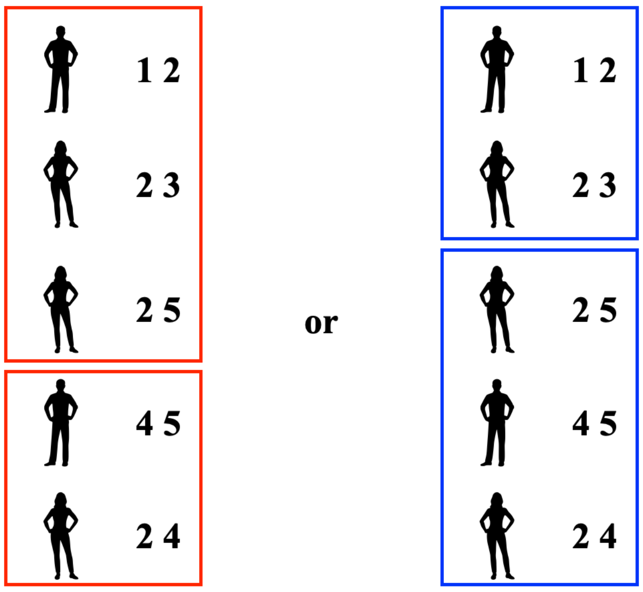


Figure 15: The above gives two different solutions to the third testcase. The solution on the right does not use any type-B group, so it is better than the one on the left.

Input

Each test case consists of $n+1$ lines. Two integers n and h are given in the first line. Then n lines follow. The $(i+1)$-th line consists of two integers x and y with $1 \le x, y \le h$, indicating that the i-th guest has hobbies x and y.

Constraints

- $2 \le n \le 10^6$.
- $3 \le h \le 2n$.

Output

Find a best solution, and output the number α of the unassigned guests and the number β of the used type-B groups in the solution. Note that our goal is to minimize α and then to minimize β while retaining α minimum possible.

Examples

standard input	standard output
2 4	2 0
1 2	
3 4	
2 3	0 0
1 2	
3 1	
5 5	0 0
1 2	
2 3	
2 4	
5 2	
5 4	

Problem L. Programmable Virus

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

A great old ancient country is developing viruses for military purposes. Currently, they have found a way to do computation in human cells. They designed a special enzyme. This enzyme has 4 binding points. One is for reading RNAs as inputs, one is for generating RNAs as outputs. One is for both reading and generating RNA back and forth to keep temporary computation states, and the final one is for reading an RNA back and forth which indicates the program commands. They need some sample programs (RNAs) for demonstration. They don't have enough time to prepare the programs. So they ask for your help.

The enzyme deals with 3 nucleobases (G,U,A,C) as a unit when reading and writing RNAs. Not all $4^3 = 64$ combinations of nucleobases are in use. They only choose 11 of them. They interpret them as numeric data, or commands.

The 11 combinations are:

Nucleobases	Value	Command
GAC	-1	
CCC	0	STOP
ACG	1	NEXT
UGA	2	PREV
UGC	3	INC
UAC	4	DEC
GCG	5	OUT
UCC	6	IN
AGG	7	BEGIN
UGU	8	END
CAC	9	(DEBUG)

Once the input binding point of an enzyme is attached to an RNA chain, the computing process starts. The enzyme moves the input binding point to the first data unit of the input RNA, moves the program binding point to the first code point of the program RNA, initializes the output binding point to empty, and generates an RNA which contains only one unit and then initializes the state binding point to it.

The enzyme executes the program commands one by one, until reaching a command called 'STOP mark'. The program commands are very simple, much different from modern programming languages. There are only 9 kinds of commands:

0 STOP mark. Stop the execution.

1 Move the state binding point to the next one. If it is after the last one, put a 0 unit.

2 Move the state binding point to the previous one. If it is before the first one, put a 0 unit.

3 Replace the data unit under the state binding point with the numerically larger one. If it goes above the largest one (9), then replace it with -1.

4 Replace the data unit under the state binding point with the numerically smaller one. If it goes below the smallest one (-1), then replace it with 9.

5 Copy the data unit under the state binding point to the output point.

6 If there are no input data units, replace the data unit under the state binding point with -1. Otherwise, replace the data unit under the state binding point with the data unit under the input point. And move the input binding point to the next data unit.

7 If the data unit under the state binding point is not 0, then move the program bind point to the next command. Otherwise, move the program binding point forward to the command after the 'matching' 8 command.

8 If the data unit under the state binding point is 0, then move the program binding point to the next command. Otherwise, move the program binding point backward to the command after the 'matching' 7 command.

The word 'matching' means to treat command 7 and command 8 as parentheses '(' and ')'. They should form a valid paired expression. And the matched commands correspond to the matched parentheses. The paired command 7 and command 8 are often used as a loop.

The program binding point moves to the next command after executing the current one, except command 0,7, and 8.

The demonstration problem is simple: Read a series of non-negative digits as a decimal number and determine if it is a multiple of k. Additionally, the program length and the running steps should not exceed 10^6.

They prepared a simulation program (simu.c) for you. The first line of the input for the simulator is the RNA program. The following lines are input RNAs for the RNA program. Use the symbol '-' in the input to represent -1, and symbols '0' to '9' to represent 0 to 9. The simulator will execute the program for each input data, and output the final result and the running step counts. The simulator additionally accepts a special command, '9'. Every time it reaches command '9', it will output the current result, the state data, and the running step counts. The execution of command '9' didn't count into the steps.

Notes:

- You can download the source code of the simulator in the attachments of DOMjudge. You can find them here.
- The simulator that used to judge the submissions won't accept command '9'.

```c
/* simu.c */
/* gcc -o simu simu.c */
#include <stdio.h>
#include <string.h>

char * cmds[10] =
    { "CCC", "ACG", "UGA", "UGC", "UAC"
    , "GCG", "UCC", "AGG", "UGU", "CAC"
    };
char code[1000002], memory[2000001], input[1000001];
static inline int get_cmd(char * code){
    for(int i=9; i>=0; --i)
        if( strncmp(cmds[i], code, 3)==0 )
            return i;
    return -1;
}
int main(){
    if( !fgets(code, 1000002, stdin) ){
        puts("no code");
        return 1;
    }
    int code_len = strlen(code);
    if( code[code_len-1]!='\n' ){
        puts("code too long");
        return 1;
    }
    code[--code_len] = 0;

    while( fgets(input, 1000001, stdin) ){
        char * ip = code, * input_p = input;
        char *mem = memory + 1000000;
        char *mem_begin=mem, *mem_end=mem;
        *mem = 0;
        int step = 0, eof = 0;
        while(1){
            if( step == 1000000 ){
                puts(" step out of bound");
                break;
            }
            if( code+code_len-ip < 3 ){
                puts(" code out of bound");
                break;
            }
            int cmd = get_cmd(ip);
            if( cmd<0 ){
                printf(" invalid code point: %c%c%c\n",
                    ip[0], ip[1], ip[2]);
                break;
            }
            if( cmd==0 ){
                puts(" stop normally");
                break;
            }
```

```
if( cmd==1 ){
    ++mem;
    if( mem>mem_end )
        *++mem_end = 0;
}
else if( cmd==2 ){
    --mem;
    if( mem<mem_begin )
        *--mem_begin = 0;
}
else if( cmd==3 ){
    if( *mem==9 )
        *mem = -1;
    else
        ++*mem;
}
else if( cmd==4 ){
    if( *mem==-1 )
        *mem = 9;
    else
        --*mem;
}
else if( cmd==5 ){
    if( *mem==-1 )
        putchar('-');
    else
        putchar('0' + *mem);
}
else if( cmd==6 ){
    while(1){
        if( !*input_p || *input_p=='\n' ){
            *mem = -1;
            break;
        }
        if( *input_p=='-' ){
            *mem = -1;
            ++input_p;
            break;
        }
        if( '0'<=*input_p && *input_p<='9' ){
            *mem = *input_p++ - '0';
            break;
        }
        ++input_p;
    }
}
else if( cmd==7 ){
    if( !*mem ){
        int skip = 1;
        while(skip){
            ip += 3;
            if( code+code_len-ip < 3 ){
                puts(" 7 can't find matched 8");
```

```
                            goto END;
                        }
                        if( strncmp(cmds[8], ip, 3)==0 )
                            --skip;
                        else if( strncmp(cmds[7], ip, 3)==0 )
                            ++skip;
                    }
                }
            }
            else if( cmd==8 ){
                if( *mem ){
                    int skip = 1;
                    while(skip){
                        ip -= 3;
                        if( ip < code ){
                            puts("8 can't find matched 7");
                            goto END;
                        }
                        if( strncmp(cmds[7], ip, 3)==0 )
                            --skip;
                        else if( strncmp(cmds[8], ip, 3)==0 )
                            ++skip;
                    }
                }
            }
            else if( cmd==9 ){
                printf("\nstep: %d, state: ", step);
                for(char *p=mem_begin; p<=mem_end; ++p)
                    if( p==mem )
                        printf("[%c]", *p<0 ? '-' : '0'+*p);
                    else
                        putchar(*p<0 ? '-' : '0'+*p);
                printf(", code:");
                for(char *p=code; p<code+code_len; p+=3)
                    printf("%c%c%c%c",
                        p==ip ? '[' : p==ip+3 ? ']' : ' ',
                        p[0], p[1], p[2]);
                if( code+code_len==ip+3 )
                    putchar(']');
                puts("");
                --step;
            }
            ++step;
            ip += 3;
        }
        END:;
    }
    return 0;
}
```

Input

The only line includes a positive integer: k.

Constraints

- $1 \leq k \leq 6$
- $1 \leq n \leq 10^9$

Output

The output file should contain one line, which is the command RNA.

This program will read a series of non-negative digits as a decimal integer n. If n is a multiple of k, output 1, otherwise output 0.

Your program reads and writes numeric data directly. The enzyme will transform them from and to nucleobases.

The RNA code length should not exceed 10^6. The number of running steps should not exceed 10^6.

Examples

standard input	standard output
1	UCCAGGUACUGUUGCGCGCCC
2	UGCAGGUACACGAGGUACUGUUGAAGGUACAC GUGCUGAUGUUCCUGCUGUACGUGCAGGUACU GAUGCACGAGGUACUGAUACACGAGGUACUGA UGCACGAGGUACUGAUACACGAGGUACUGAUG CACGAGGUACUGAUACACGAGGUACUGAUGCA CGAGGUACUGAUACACGAGGUACUGAUGCACG AGGUACUGAUACACGUGUUGUUGUUGUUGUUG UUGUUGUUGUUGUUGAGCGCCC
3	UGCAGGACGACGUCCAGGUACACGUGCACGUG CUGAUGAUGUACGACGACGUGCUGAUGCAGGU ACACGAGGUACUGUUGAUGUACGAGGUGAUGA UGAUGCUGAAGGUACACGAGGUACUGUUGAUG UACGGCGCCCUGUUGAUGAAGGUACUGAUGCA CGAGGUACUGAUGCACGAGGUACUGAUACUAC ACGAGGUACUGAUGCACGAGGUACUGAUGCAC GAGGUACUGAUACUACACGAGGUACUGAUGCA CGAGGUACUGAUGCACGAGGUACUGAUACUAC ACGUGUUGUUGUUGUUGUUGUUGUUGUUGUUG AAGGUACUGAUGCACGUGUUGAAGGUACACGU GCUGAAGGUACACGUGCUGAAGGUACACGUAC UACUGAAGGUACACGUGCUGAUGUUGUUGUUG UACGAGGUACUGAUGCACGUGUUGAUGAUGU
6	UGCAGGACGACGUCCAGGUACACGUGCACGUG CUGAUGAUGUACGACGACGUGCUGAUGCAGGU ACACGAGGUACUGUUGAUGUACGAGGUGAUGA UGAUGAAGGAGGUACUGUGCGCCCUGUACGAC GACGACGACGACGUGCAGGUACUGAUGCACGA GGUACUGAUACACGAGGUACUGAUGCACGAGG UACUGAUACACGAGGUACUGAUGCACGAGGUA CUGAUACACGAGGUACUGAUGCACGAGGUACU GAUACACGAGGUACUGAUGCACGAGGUACUGA UACACGUGUUGUUGUUGUUGUUGUUGUUGUUG UUGUUGAGCGCCCUGUUGAACGACGAGGUACU GUACGAGGUACUGUUGAUGAUGAUGAAGGUAC ACGACGACGUGCACGUGCUGAUGAUGAUGAUG UACGACGACGAGGUACUGAUGAUGAUGCACGA CGACGUGUUGAUGAUGAAGGUACUGAUGCACG AGGUACUGAUGCACGAGGUACUGAUACUACAC GAGGUACUGAUGCACGAGGUACUGAUGCACGA GGUACUGAUACUACACGAGGUACUGAUGCACG AGGUACUGAUGCACGAGGUACUGAUACUACAC GUGUUGUUGUUGUUGUUGUUGUUGUUGUUGAA GGUACUGAUGCACGUGUUGAAGGUACACGUGC UGAAGGUACACGUGCUGAAGGUACACGUACUA CUGAAGGUACACGUGCUGAUGUUGUUGUUGUA CGAGGUACUGAUGCACGUGUUGAUGAUGU

Notes

There's only one line in each sample output. We broke it into lines in samples because of the paper layout. You should concatenate them into one line.

The following is the running example of the first sample program.

Program	Memory	Input	Output	Comment
UCC AGG UAC UGU UGC GCG CCC		3		
UCC AGG UAC UGU UGC GCG CCC	3			IN
UCC AGG UAC UGU UGC GCG CCC	3			BEGIN: 3 != 0
UCC AGG UAC UGU UGC GCG CCC	2			DEC
UCC AGG UAC UGU UGC GCG CCC	2			END: 2 != 0
UCC AGG UAC UGU UGC GCG CCC	1			DEC
UCC AGG UAC UGU UGC GCG CCC	1			END: 1 != 0
UCC AGG UAC UGU UGC GCG CCC	0			DEC
UCC AGG UAC UGU UGC GCG CCC	0			END: 0 == 0
UCC AGG UAC UGU UGC GCG CCC	1			INC
UCC AGG UAC UGU UGC GCG CCC	1		1	OUT
UCC AGG UAC UGU UGC GCG CCC	1		1	STOP

Problem M. Connectivity Problem

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Captain Jack occupies n islands. He wants to build a kingdom in the n islands. He names each island with an index i, $1 \leq i \leq n$. At the beginning, there was no bridge between any two islands. However, Jack wants to inspect any two islands p and q by car. When Jack cannot travel from island p to island q, he will build a bridge between islands p and q. Given two island indices p and q, please write a program to help Jack to see whether he can travel from p to q. For example, at first Jack wants to travel from island 2 to island 3, but there is no bridge between them. Then the program must output "N" because he cannot travel from island 2 to island 3 by car. After that, Jack decides to build a bridge between island 2 and island 3. Next, Jack wants to travel from island 3 to island 4, but there is still no bridge connecting island 4 to any other islands that are connected to island 3. So your program must output "N". Next, Jack wants to travel from island 2 to island 4, your program will output "Y" because there is a bridge between islands 2 and 3 and there is a bridge between 3 and 4. In Table 2, we give an example of this process.

Table 2: Connectivity example

Input p q	Output	Previous pairs imply that p is connected to q
2 3	N	Build a bridge between island 2 and island 3
3 4	N	Build a bridge between island 3 and island 4
2 4	Y	There is a bridge between islands 2 and 3 and there is a bridge between 3 and 4.

Input

The first line of input is an integer n, $1 \leq n \leq 10000$, denoting the number of times that Jack travels between two islands. The following n lines contains two integers p and q separated with a space indicating that Jack wants to travel from island p to island q where $0 \leq p, q < 1000$.

Constraints

- $1 \leq n \leq 10000$
- $0 \leq p, q < 1000$, and $p \neq q$.

Output

For each test case, output one line containing "Y" or "N" which indicates whether Jack can travel from island p to island q or not.

Examples

standard input	standard output

12
3 4
4 9
8 1
2 3
5 6
2 9
5 9
7 3
4 8
5 6
1 8
6 1
	N
	N
	N
	N
	N
	Y
	N
	N
	N
	Y
	Y
	Y