The 2nd Universal Cup

Stage 21: Delft

February 3-4, 2024

This problem set should contain 11 problems on 15 numbered pages.

Based on

Osijek Competitive Programming Camp

Problem A. Anton's ABCD

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

It is well-known that Anton Trygub likes setting problems about strings with the characters `ABC` in them. And he likes it even more when we are allowed to do operations on the string. To honour him, but also to improve further on his problems, here is a problem about a string with the characters `ABCD` in it. You are given a string consisting of the characters `ABCD`. How many distinct strings can you make, applying the following operation zero or more times?

- Take a substring of length 4, that is a cyclic shift of `ABCD`, and cyclic shift it to the left or right by 1. (The cyclic shifts of `ABCD` are `BCDA`, `CDAB` and `DABC`.)

Because this number can be huge, please output it modulo $10^9 + 7$

Input

The first and only line of input contains the string S, $(1 \leq |S| \leq 2000)$

Output

Print a single line of output with one integer, the number of distinct strings that can be made using the operation repeatedly, modulo $10^9 + 7$.

Examples

standard input	standard output
DABC	4
AABBCCDD	1
ABCDABCD	52

Note

In the first example, the string is a cyclic shift of `ABCD` itself, and so within 2 operations, we can get all cyclic shifts of this string, of which there are 4.

In the second example, no move is possible in the beginning, so there's only one distinct string you can make.

Problem B. Beer Circuits

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	256 megabytes

There are n pubs in the beautiful city of Delft, marked on a two-dimensional map as points (the pubs are far enough apart that this simplification makes sense). A common thing for college freshmen is to do a Beer Circuit, where they plan out a route to go to some pubs and drink a beer in each one.

They decide on some order of their selected pubs, visit the pubs one after another, drink a beer in each one, and after visiting all pubs in their circuit return back to the starting pub. It is boring to only visit 2 pubs, so it is customary to visit at least 3 pubs, but not more than k different pubs can be in a route.

After having multiple beers, walking for a long time is tiring. So the Beer Circuit organizers want to make a circuit, such that the longest distance (euclidean distance) between two adjacent pubs in the circuit, is as small as possible. Note that after having a beer in the next pub, the students feel refreshed, so the total walking distance is not super important.

After minimizing this maximum distance between two adjacent pubs in the Beer Circuit, over all Beer Circuits that are left, the organizers decide to only pick Beer Circuits with the least amount of pubs in them (of course this amount is not smaller than 3).

How many options do the organizers have for choosing their desired Beer Circuit? Two Beer Circuits are different if they visit a different set of pubs or the order of visiting the pubs is different. Note that if two Beer Circuits starts at a different starting pub but are otherwise the same, they are also considered different.

Input

The first line of input contains two integers, n and k ($3 \leq n \leq 200\,000$, $3 \leq k \leq 30$) — the number of points in the input and the maximum size of a Beer Circuit.

Each of the next n lines contains the coordinates of a pub. Each line contains two integers x and y ($0 \leq x, y \leq 10^9$) — the coordinates of the pub, as marked on the map.

It is guaranteed that no two pubs are in the exact same location.

Output

On the first line, print a single integer, the minimum possible euclidean distance squared of the longest adjacent distance in a Beer Circuit. The squared euclidean distance can be calculated using the formula $(\Delta x)^2 + (\Delta y)^2$

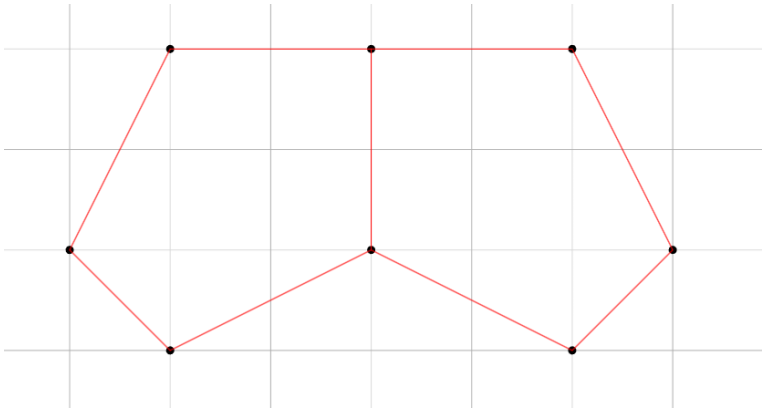
On the second line, print another integer, after minimizing the maximum adjacent euclidean distance of a Beer Circuit, what is the minimum number of pubs in such a Beer Circuit.

On the third line, print the number of Beer Circuits, that fulfill all the criteria of the organizers.

Examples

standard input	standard output
3 3 0 0 2 2 1000000000 1000000000	2000000000000000000 3 6
8 5 5 5 5 7 7 7 3 7 2 5 8 5 3 4 7 4	5 5 20

Note



Visualization of example 2. The red edges show the two Beer Circuits which the organizers want. The circuits both use the middle 2 pubs, but one traverses the left halve of the pubs, and the other circuit uses the right halve. They are of length 5. Each one can be traversed in 10 different orders, so in total this gives 20 different possibilities.

Note that in example 2, there is also a Beer Circuit which uses all 8 vertices, and has the same maximum euclidean distance, but the number of pubs in it is not the minimum possible.

Problem C. Curly Palindromes

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

You are given n points in the plane ($n \leq 100$). All the points are distinct. Each point is labeled with a letter.

Find the largest curly palindrome.

A *curly palindrome* is a sequence such that:

- for all $2 \leq i \leq \text{len}(q) - 1$, the angle of the points $q_{i-1}q_iq_{i+1}$, is counterclockwise. Formally, let $\mathbf{u} = q_i - q_{i-1}$ and $\mathbf{v} = q_{i+1} - q_i$, then $\text{cross}(\mathbf{u}, \mathbf{v}) > 0$, $\text{cross}(\mathbf{a}, \mathbf{b}) := \mathbf{a}_x \cdot \mathbf{b}_y - \mathbf{a}_y \cdot \mathbf{b}_x$.
- No two adjacent points in the sequence should be equal, $q_i \neq q_{i+1}$. It is allowed to use the same point multiple times, as long as the occurrences are not adjacent.
- Lastly, the labels on the points in the curly palindrome must spell out a palindrome.

If you can make curly palindromes of unbounded size, output "`Infinity`".

Input

The first line of input contains a single integer, n ($1 \leq n \leq 100$) — the number of points in the input.

Each of the next n lines contains the description of a labeled point. Each line contains two integers x and y and a lowercase english character c, ($0 \leq x, y \leq 10^9$) — the coordinates of the point and the label of the point.

Output

Print a single line containing the length of the largest curly palindrome of the labeled pointset. If the length can be unbounded, instead print on a single line the word "`Infinity`"

Examples

standard input	standard output
4 0 0 o 1 1 c 2 2 p 3 3 c	2
3 2 3 e 3 2 e 8 9 e	Infinity

Note

In the second case, palindromes of any length k can be made, by starting from some point, and repeatedly walking counterclockwise around the triangle formed by the three e's. This length can be unbounded, so the output is "`Infinity`"

Problem D. Division 3 Polyglot

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

A polyglot is a computer program that can be successfully compiled in different languages while still behaving the same. But what would a polyglot testcase in competitive programming look like? Let's take two random Division 3 problems from Codeforces:

1 Problem X: Karina and Array

Karina has an array of n integers $a_1, a_2, a_3, \ldots, a_n$. She loves multiplying numbers, so she decided that the beauty of a pair of numbers is their product. And the beauty of an array is the maximum beauty of a pair of adjacent elements in the array.

For example, for $n = 4$, $a = [3, 5, 7, 4]$, the beauty of the array is $\max(3{\cdot}5, 5{\cdot}7, 7{\cdot}4) = \max(15, 35, 28) = 35$.

Karina wants her array to be as beautiful as possible. In order to achieve her goal, she can remove some elements (possibly zero) from the array. After Karina removes all elements she wants to, the array must contain at least two elements.

Unfortunately, Karina doesn't have enough time to do all her tasks, so she asks you to calculate the maximum beauty of the array that she can get by removing any number of elements (possibly zero).

1.1 Input

The first line of the input contains an integer t ($1 \le t \le 10^4$) — the number of test cases.

The description of the test cases follows.

For each test case:

- The first line contains an integer n ($2 \le n \le 2 \times 10^5$) — the length of the array a.
- The second line contains n integers $a_1, a_2, a_3, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$) — the elements of the array a.

The sum of all values of n across all test cases does not exceed 2×10^5.

1.2 Output

Output t integers, each of which is the answer to the corresponding test case — the maximum beauty of the array that Karina can get.

2 Problem Y: Boats Competition

There are n people who want to participate in a boat competition. The weight of the i-th participant is w_i. Only teams consisting of two people can participate in this competition. As an organizer, you think that it's fair to allow only teams with the same total weight.

So, if there are k teams $(a_1, b_1), (a_2, b_2), \ldots, (a_k, b_k)$, where a_i is the weight of the first participant of the i-th team and b_i is the weight of the second participant of the i-th team, then the condition $a_1 + b_1 = a_2 + b_2 = \ldots = a_k + b_k = s$, where s is the total weight of each team, should be satisfied.

Your task is to choose such s that the number of teams people can create is the maximum possible. Note that each participant can be in no more than one team.

2.1 Input

The first line of the input contains one integer t ($1 \le t \le 1000$) — the number of test cases. Then t test cases follow.

For each test case:

- The first line contains one integer n ($1 \le n \le 50$) — the number of participants.
- The second line contains n integers $w_1, w_2, \ldots, w_n$ ($1 \le w_i \le n$), where w_i is the weight of the i-th participant.

2.2 Output

For each test case, print one integer k: the maximum number of teams people can compose with the total weight s, if you choose s optimally.

3 Actual Problem

Now, you are given an integer x, and your job is to make a testcase that is valid for both problems above, and such that a correct solution for both problems would output the single integer x.

Input

The first and only line contains an integer x ($1 \le x \le 25$) — the desired output of the correct solutions to the CodeForces problems.

Output

On multiple lines, output a valid testcase. This testcase should be valid input to both CodeForces problems, and a correct solution for either problem should output a single line with the integer x.

Example

standard input	standard output
6	1 14 2 1 2 3 1 2 1 1 2 1 2 2 1 2

Problem E. Enumerating Substrings

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

There's an alphabet of size k. For a string S in this alphabet (the text), and string P (the pattern), let $F(S, P)$ = the maximum number of non-overlapping substrings you can take in S, that are equal to P.

Let's call a string Q *beautiful*, if each letter in it occurs no more than 2 times.

Over all possible strings of size n, and all possible beautiful patterns P of size m, calculate the sum of $F(S, P)$. Because this sum can be huge, output the result modulo $10^9 + 7$.

Input

The first and only line of the input contains 3 integers, n, m, k ($1 \le n \le 10^6$, $1 \le m \le 2000$, $m \le n$ and $1 \le k \le 10^9$) — respectively, the length of string S, the length of the pattern P and the alphabet size.

Output

Print a single line, containing one integer — the sum of $F(S, P)$ over all strings S and beautiful strings P modulo $10^9 + 7$.

Examples

standard input	standard output
4 2 3	228
999999 1999 12345678	52352722

Problem F. Football in Osijek

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

The famous Osijek Competitive Programming Camp has grown so popular, it branched out to other training camps as well, one of which is the Osijek Football Training Camp. The big Osijek Football Cup is soon starting, so the camp wants to assemble a team. But the rules are quite strict and require that a team consists of exactly k players. The camp has n football players, numbered from 1 to n. The players have preferences for who they want to be in the team with. Player i has a preference of the form:

"If I join the team, then player a_i must also join the team."

It can be that $i = a_i$; that means that this particular player only cares about themselves.

Further more, the players have an extra demand:

"I don't want to be in a team with somebody who is not connected to me with a *preference chain*."

A *preference chain* is a sequence of players $u, p_2, p_3, p_4 ... v$, such that the next player in the sequence is $a_{\text{current player}}$. Two players u and v are connected, if there's a preference chain from u to v, or from v to u.

Unfortunately, after careful reading of the rules, it's still vague what the value of k is. So for all k from 1 to n you want to consider the scenario, where you need a team of size k. If it's impossible to make a team, you decide to talk to the players. In one meeting, you can choose a player i, and convince them to change their preference. Formally, you choose $1 \le i, x \le n$ and do the update $a_i := x$. After this it's allowed that $a_i = i$.

Find, for each k from 1 to n, the least number of meetings you need to form a team of size k.

Input

The first line of input contains one integer, n ($1 \le n \le 500\,000$) — the number of players in the Osijek Football Training Camp.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the current preferences of all the players.

Output

Output one line consisting of n integers, the minimum number of meetings for all k from 1 to n.

Example

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

Problem G. Graffiti

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Jeroen has found a beautiful tree consisting of n nodes. Help him put graffiti on the tree. Jeroen applies graffiti to the tree by spraypainting a letter on each node. He has a favourite word and he wants to maximize the number of directed paths in the tree $u \to v$, such that the letters on the path $u, p_2, p_3, \ldots, v$, spell out his favourite word.

Note that a directed path is a sequence of 1 or more nodes, where adjacent nodes in the sequence are neighbours in the tree, and no node appears more than once in the sequence. The tree itself is undirected, so each edge can be traversed in both directions.

Input

The first line of the input contains a single integer n ($1 \le n \le 300\,000$) — the number of nodes in the tree.

The next line contains a string w ($1 \le |w| \le 3$) consisting only of lowercase letters — Jeroen's favourite word.

Each of the next $n-1$ lines contains the description of an edge. Each line contains two integers u_i and v_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$) — an edge connects node u_i to node v_i.

It is guaranteed that the $n-1$ edges form a tree.

Output

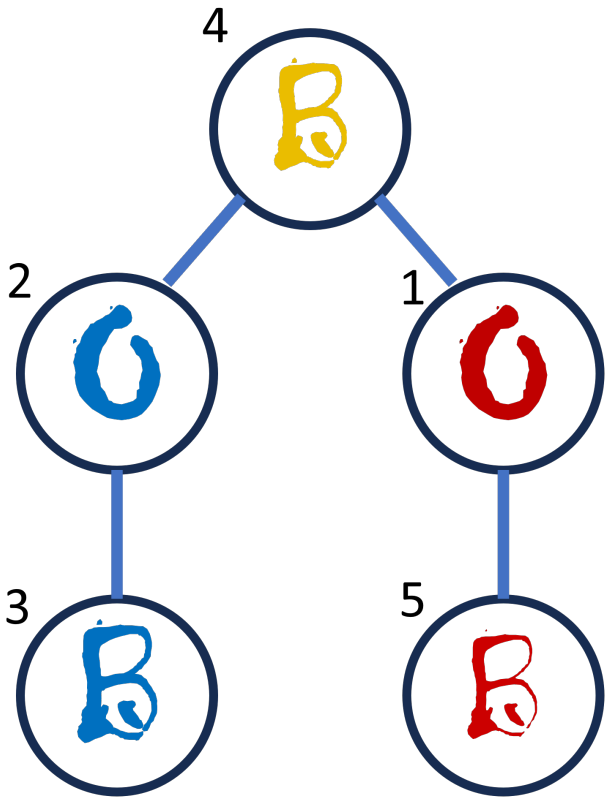
Output a single integer, the maximum number of directed paths that spell out Jeroen's favourite word, if you spraypaint the letters optimally.

Examples

standard input	standard output
1 a	1
3 orz 1 2 2 3	1
2 ab 1 2	1
5 bob 3 2 5 1 1 4 2 4	4

Note

For the last sample, one way to optimally spraypaint the letters is:



Visualization of example 4

Here the directed paths that contain the word `bob`, are:

- $4 \to 1 \to 5$
- $5 \to 1 \to 4$
- $4 \to 2 \to 3$
- $3 \to 2 \to 4$

In total, this gives 4 directed paths, with the word `bob`, which is optimal.

Problem H. Huge Oil Platform

Input file:	`standard input`
Output file:	`standard output`
Time limit:	8 seconds
Memory limit:	256 megabytes

Jeroen is planning to build an oil platform in the North Sea. He has found n ($n \leq 400$) locations, where oil can be extracted.

For each place, he has estimated the profit he would get if the oil platform could extract this oil. Because of the hugeness of the oil platform, the oil locations can be seen as points in the plane.

For the purposes of this problem, the earth can be considered flat.

Because of regulations, an oil rig has to be in the shape of a rectangle. Further regulations also make your life difficult. The oil rig itself is quite cheap, but the security fences around the outside of the oil rig are very expensive. This means the cost of building an oil rig is equal to the perimeter of this rectangle. All the locations inside or on the border of the oil platform can be extracted from. It is allowed to build degenerate oil platforms, where the width and/or the height are 0.

What is the maximum possible profit Jeroen can get (that is the profits from the oil extraction minus the cost of building the oil platform)? Your answer is considered correct if its absolute or relative error is less than 10^{-9}.

Input

The first line of input contains a single integer, n ($1 \leq n \leq 400$) — the number of candidate locations.

Each of the next n lines contains the description of a candidate drilling location. Each line contains three integers x, y and w, ($0 \leq x, y \leq 10^6$, $1 \leq w \leq 10^9$) — the coordinates of the point and the profit that can be made drilling at this location.

It is guaranteed that the candidate locations are all distinct.

Output

Print the maximum profit obtainable with a rectangular oil rig. The result should have a relative or absolute error of at most 10^{-6}.

Formally, let your answer be a, and the jury's answer be b. Your answer is accepted if and only if

$$\frac{|a-b|}{\max(1,|b|)} \leq 10^{-6}$$

Examples

standard input	standard output
2 1 1 1 3 3 1	1
3 4 5 5 4 6 7 1 3 8	10.1005050633883
2 0 0 1 1000000 1000000 1000000000	1000000000

Problem I. Isomorphic Delight

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Given a number n, create a simple undirected graph on n nodes, that is asymmetric and has the least number of edges, or output that no such graph exists.

A graph is asymmetric if there are no relabelings of the vertices (except the identity permutation), such that you obtain exactly the same graph.

Formally: For a graph (V, E) to be asymmetric, there should **not** exist a permutation π of the vertices, such that π is not the identity permutation, and it holds that: $uv \in E \Leftrightarrow \pi(u)\pi(v) \in E$.

Input

The first and only line in the input contains one integer n ($1 \leq n \leq 10^6$) — the number of nodes the graph should have.

Output

Output "`YES`" if there exists an asymmetric graph with n nodes, otherwise print "`NO`". If the answer is "`YES`", on the following lines output a description of such a graph with the lowest number of edges.

The first line of the description is a single integer m, the number of edges in your graph. Each of the next m lines should contain 2 integers u and v, denoting an undirected edge between nodes u and v. No undirected edge should appear more than once in the output (otherwise the graph is not simple), and the graph should be asymmetric.

Examples

standard input	standard output
1	YES 0
6	YES 6 1 2 2 3 1 3 3 4 2 5 5 6
4	NO

Problem J. Jumbled Primes

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	256 megabytes

The great oracle has chosen a uniformly random permutation of the numbers from 1 to 100: $p_1, p_2, \ldots, p_{100}$.

The only way to get information from the oracle is by asking for the greatest common divisor (gcd) of a pair of numbers in the permutation, p_a and p_b, $(a \neq b)$.

Find all the positions of the prime numbers and the number one, and report them. The output should be a bitstring with 1 if position i is a prime or the number one, 0 otherwise.

Your solution is run once, and in this run, it will be tested on a 1000 testcases. Note that each time you submit, the tests are newly generated. Let S be the sum of the number of queries used on all tests. For getting accepted, it must hold that: $S \leq 1000 \times 600$

Interaction Protocol

In a testcase, your program can immediately start doing queries.

For finding out the gcd of p_a and p_b, output a single line in the format "`? a b`", where a and b are two distinct integers, such that $1 \leq a, b \leq 100$. Afterwards, you should read a single integer g. g will be equal to $\gcd(p_a, p_b)$.

When you have determined all the positions of the prime numbers and the number one, print a single line of the form "`! answer`", where `answer` is a bitstring of length 100, with a 1 at position j if p_j is a prime or the number one. When p_j is composite, the bitstring should contain a 0 at that position. This does not contribute to the total query count S.

Afterwards, this testcase is finished, and your program should immediately move on to the next testcase, or stop, if it completed all 1000 tests.

Do not forget to flush the output after each query.

Example

standard input	standard output
1 2 2 3	? 1 3 ? 2 4 ? 6 4 ? 6 3 ! 111010

Note

This example is not a valid testcase. For illustration, here p is a permutation of the numbers from 1 to 6. The random permutation just happened to be $p = [1\ 2\ 3\ 4\ 5\ 6]$.

Although the information obtained by the queries is not enough to find out where the primes are located, the output is correct: A bitstring of length 6, with a 1 at places $1, 2, 3$ and 5. p_1, p_2, p_3 and p_5 contain the prime numbers and one.

Problem K. King's Dinner

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

The King of the Higherlands has organized a big dinner where he invited the whole royal family. The event will take place in a big square hall of $n \times n$ meters, divided into $n \times n$ squares. The king wants to invite as many guests as possible, but for each guest there needs to be space for a table. The tables in question are rectangular with dimensions of 1 meter by 2 meter. According to royal rules, any table should cover any two squares of the hall completely. To have enough room for the guests to sit, tables should not be adjacent. Of course, tables cannot be stacked on top of eachother.

Two grid squares are adjacent if they share an edge or a corner. Two tables are adjacent if any of their squares are adjacent.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 50$). The description of the test cases follows.

Each test case consists of one line. This line contains a single integer n ($n \le 100$) — the size of the hall.

It is guaranteed that the sum of n^2 over all testcases does not exceed 200 000

Output

For each testcase, print any $n \times n$ grid with the maximum number of tables, that are not stacked on top of each other and are not adjacent. In the grid, print a '`#`' if the square is covered by a table, and print a '`.`' if the square is not covered.

Print the $n \times n$ grid on n lines of n characters each. If there are multiple solutions, print any.

Example

standard input	standard output
3 1 2 3	. #. #. #.# #.# ...

Note

In the first sample, the hall is only 1×1 meter, so not even a single table fits.

In the second sample, 1 table can be placed in the hall. It can be proven that 2 tables or more is impossible.

In the third sample, a hall with 2 tables is shown, and that is also optimal.