Problem A
Octal Equivalence
Problem ID: octal
Time Limit: 1

In some programming languages, integer literals that begin with a 0 are actually octal literals (*i.e.*, base 8). For example, consider the following C++ code.

```
if (monthName == "August")
    monthValue = 008;
else if (monthName == "December")
    monthValue = 012;
```

This will result in `monthValue` having the correct value of decimal 8 in the case of August, but the *incorrect* value of decimal 10 (because $1 \times 8^1 + 2 \times 8^0 = 10$) in the case of December!

We could debate the sensibility of this feature from a language design perspective, but for now, let us just focus on checking our program for possible errors. Assuming we always mean to express our values in decimal, do the literals in our program represent the correct values even though the compiler will interpret them as octal numbers?

Input

The first line of input contains a single positive integer $n \leq 100$, denoting the number of test cases. This is followed by n lines, each containing one integer in octal notation, which will have at least one leading zero (0) followed by one or more digits. Each integer will have fewer than 100 digits.

Output

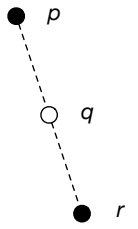
For each test case, print `equivalent` if the octal number, when interpreted as a decimal number, has the same value. Otherwise, print `not equivalent`. The output for each test case should be on its own line.

Sample Input	Sample Output
3 007 012 00000000000000000000000000000000000000000000000012	equivalent not equivalent not equivalent

This page is intentionally left blank.

Problem B
Point Negation
Problem ID: negation
Time Limit: 5

Let p and q be two points in the plane. The **negation** of p about q is the point r such that q is the midpoint between p and r. If $p = q$, then $r = p = q$ as well.



Say that a set of points P has **negation symmetry** if there is some point q such that for every point $p \in P$, the negation of p about q is also in P.

Input

The first line consists of a single integer n between 1 and $100,000$. Then n lines follow, each describing a point p_i as a pair x_i, y_i with $-10^9 \leq x_i, y_i \leq 10^9$. These n points are guaranteed to be distinct.

Output

Output a single line with the message `symmetric` or `not symmetric`, indicating if the given set of n points has negation symmetry or not.

Sample Input	Sample Output
6 3 2 3 0 10 -7 1 0 1 2 -6 9	symmetric

Sample Input	Sample Output
4 1 1 4 4 3 3 2 1	not symmetric

Sample Input	Sample Output
3 -1 0 0 0 1 0	symmetric

Problem C
Card Game
Problem ID: cardgame
Time Limit: 10

Your friend Charlie is good at playing a certain solitaire card game: In it we are given a sequence of $n + 2$ cards with values $v_0, \ldots, v_n, v_{n+1}$ where $v_0 = v_{n+1} = 1$. Cards 0 and $n + 1$ are marked as **unplayable**.

A turn consists of picking a card i that is not marked unplayable. The player then earns $v_{i-1} \cdot v_i \cdot v_{i+1}$ points (which may be negative). The value of card v_i is then set to 1 and v_i is also marked unplayable.

Charlie can quit at any time, not all cards need to be marked unplayable. What is the maximum amount of points that Charlie can earn?

Input

The first line consists of a single integer n. This followed by a containing the numbers $v_1, v_2, \ldots, v_n$ for the n cards that are initially playable. You are guaranteed that $1 \leq n \leq 300$ and $-100 \leq v_i \leq 100$.

Note, v_0 and v_{n+1} are understood to be 1 and are not included in the input.

Output

Output a single value indicating the maximum points Charlie can earn.

Sample Input	Sample Output
4 3 1 5 8	66

Sample Input	Sample Output
3 -4 -1 -7	11

Sample Input	Sample Output
4 100 -100 0 100	200

This page is intentionally left blank.

Problem D
Independent Sets of Friends
Problem ID: indsets
Time Limit: 3

It's nearing the end of the term. You want to hold a big party!

You have many friends to invite. But some of them do not get along with each other. The party should be a fun and peaceful time, so you will only invite a group of friends if every pair of invited friends gets along.

Invite as many friends as possible! If it is not possible to invite most of your friends, then you might as well not throw the party.

Input

The first line of input consists of three integers n, m, k where $1 \leq n \leq 5,000, 0 \leq m \leq 200,000$ and $\max\{0, n-18\} \leq k \leq n$. Here, n is the number of friends in the graph, m the number of pairs of friends that do not get along, and k is the minimum number of friends you need to invite to consider the party worth holding. Your friends are numbered from 0 through $n-1$.

Then m lines follow, each describing an pair of friends who do not get along by giving the indices i, j of the friends (you are guaranteed $i \neq j$). Each pair of friends who do not get along appears at most once, so if some line has i, j then no other line has i, j or j, i.

Output

If it is impossible to invite at least k friends to the party such that no two of the friends fight, then output a single line with the message `not big enough`. Otherwise, output the largest integer k' such that it is possible to have a party with k' friends where no two of them fight.

Sample Input	Sample Output
5 4 4 0 1 0 2 0 3 0 4	4

Sample Input	Sample Output
5 5 4 0 1 0 2 0 3 0 4 1 4	not big enough

Sample Input	Sample Output
6 5 2 0 1 2 3 3 4 2 4 3 5	3

Problem E
RSA Mistake
Problem ID: rsamistake
Time Limit: 5

An RSA number n is a composite number that is the product of two distinct odd prime numbers p, q. Such numbers are used in the RSA public key cryptosystem.

Johnny is working on his crytography assignment and needs to trace the execution of the algorithm that performs secure encryption. He generates two distinct odd prime numbers p, q and multiplies them to form a number n and proceeds to finish the rest of his assignment.

Unfortunately, Johnny may have made a mistake. He is worried he formed n incorrectly: that he multiplied one of the primes into n twice. Johnny also forgot the original primes! Since factoring integers is challenging, Johnny needs your help.

Input

Input consists of a single integer $15 \leq n < 2^{64}$. You are guaranteed that n is either of the form $p \cdot q$ or $p^2 \cdot q$ where p and q are distinct odd primes.

Output

If $n = p \cdot q$ for distinct odd primes p, q, output a single line with the message `RSA OK`. Otherwise, if $n = p^2 \cdot q$ for distinct odd primes p, q, output a single line with the message `RSA MISTAKE`.

The output for the last example is `RSA MISTAKE` because $n = p^2 \cdot q$ where $p = 2642239$ and $q = 2642257$.

Sample Input	Sample Output
33	RSA OK

Sample Input	Sample Output
63	RSA MISTAKE

Sample Input	Sample Output
18446724184027494097	RSA MISTAKE

This page is intentionally left blank.

Problem F
Set Your Clocks Properly!
Problem ID: clockadjust
Time Limit: 1

Daylight saving time began last week! Hopefully, you managed to set all your clocks properly without any problems. Sometimes, it's not the easiest thing to do...

Some digital clocks only offer three buttons for setting the time. In this problem, we'll call them `S`, `U`, and `D`, meaning SET, UP, and DOWN, respectively. The function of the SET button (`S`) is to cycle the clock between display-mode (for simply displaying the time), hour-setting-mode (for changing the hour on the clock), and minute-setting-mode (for changing the minute of the clock). In other words, a clock that is in display-mode will change to hour-setting-mode after pressing SET once, and then will change to minute-setting-mode after pressing SET again, and then back to display-mode after pressing SET one more time. The UP (`U`) and DOWN (`D`) buttons don't do anything in display-mode, but in the other modes they increment or decrement the hour (in hour-setting-mode) or minute (in minute-setting-mode) by one. Note this is a simple 12-hour clock with no indication of AM or PM, and no counter for the seconds. The hour can be any value from 1 to 12 inclusive, and the minute can be any value from 00 to 59 inclusive. Of course, the UP and DOWN buttons wraparound, so that incrementing or decrementing past the valid range will simply wraparound to the other side of the range.

Phew! That's complicated, isn't it? Since we're programmers, let's write a program to help us figure out how to press these buttons. Given a clock in display-mode, the current time, and a target time, can you output the sequence that makes the clock display the target time and returns it to display-mode in the *fewest* button presses? If there exists more than one such shortest sequence, output the one with the *most* UP presses.

Input

Input will consist of two lines. The first line will be the currently shown time in the format `HH:MM`, where `HH` is an integer (possibly with a leading zero, $01 \leq$ `HH` ≤ 12) that denotes the current hour, and `MM` is an integer (possibly with a leading zero, $00 \leq$ `MM` ≤ 59) that denotes the current minute. The second line gives the target time in a similar fashion. It is guaranteed that the current time and target time are not identical.

Output

Print on a single line a string consisting of characters `S`, `U`, or `D`, which gives the shortest possible sequence that will make the clock show the target time and return it to display-mode. Again, if there exists more than one such shortest sequence, output the one with the *most* UP presses.

Sample Input	Sample Output
02:00 03:00	SUSS

Sample Input	Sample Output
10:30 10:00	SSUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUS

Sample Input	Sample Output
09:14 03:11	SUUUUUUSDDDS

Problem G
Standoff
Problem ID: standoff
Time Limit: 2 seconds

In the land of Cartexico, the citizens are all gun-toting quick-drawing cowboys who can only shoot in the four cardinal directions (*i.e.*, north, south, east, west). The citizens are also quite confrontational, so they often find themselves in tense situations in which multiple cowboys are threatening to shoot one another. Such situations are known to those outside of Cartexico as a Cartexican standoff.

Cartexican standoffs resolve according to the following rules:

- Cowboys can shoot only in the four cardinal directions, and cannot hit cowboys that are directly behind other cowboys (*i.e.*, cowboys block line-of-sight). In other words, if there are three cowboys lined up north to south, the northernmost cowboy can only hit the middle cowboy, the southernmost cowboy can only hit the middle cowboy, while the middle cowboy can hit both of the other cowboys.
- Once a cowboy is shot, they immediately fall to the ground and no longer block line-of-sight. In the lined up example, this means that the northernmost cowboy could shoot the middle cowboy, and then shoot the southernmost cowboy.
- Cowboys cannot shoot themselves.
- Bullets have no travel time, and a cowboy cannot shoot after being shot. No two cowboys will ever shoot at exactly the same time. In other words, if there are exactly two cowboys able to shoot each other, only one of them can be shot, because one will get the shot off first, rendering the other unable to shoot.
- Once the cowboys start shooting, shots continue to be fired until the remaining cowboys are unable to shoot one another.
- Cowboys have unlimited bullets.

You work at the regional medical centre, which is responsible for dispatching ambulances to the scene after such standoffs are resolved. You have access to satellite images that give you the cowboy positions for any standoff. Furthermore, you have comprehensive psychological profiles on every citizen of Cartexico, so you can predict with 100% certainty which cowboy will shoot first, and in which direction. Unfortunately, after this first shot is fired, chaos ensues, so the order and direction in which the remaining cowboys shoot is difficult to predict. Since resources are scarce, any programs that can help you allocate these resources will be very valuable. In particular, you want to know, for a given standoff and a given initial shot, what is the *maximum* number of ambulances that you might need to deploy after the standoff resolves?

Input

The first line of input contains a single positive integer $n \leq 10,000$, denoting the number of cowboys in the standoff. This is followed by n lines, each containing two integers x_i and y_i, denoting the coordinates of the i^{th} cowboy. (Of course, since this is the land of Cartexico, each cowboy exists at integer coordinates, $-2^{30} \leq x_i, y_i \leq 2^{30}$.) This is followed by one line, $x\ y\ dir$, where dir is one of $north$, $east$, $south$, $west$, indicating that the standoff resolves starting with the cowboy at (x, y) shooting in the direction dir. It is

guaranteed that a cowboy does exist at (x, y), and that a target cowboy exists in the direction of the initial shot (*i.e.*, the first shot is always made towards another cowboy).

Output

Output the maximum number of cowboys that could need medical attention after the standoff resolves (*i.e.*, the maximum number of cowboys that could be shot during the resolution of the standoff that begins with the indicated first shot).

Sample Input	Sample Output
4 0 0 -10 0 10 0 0 -10 0 0 west	3

Sample Input	Sample Output
4 0 0 -10 0 10 0 0 -10 -10 0 east	2

Sample Input	Sample Output
10 0 0 1 0 2 0 4 0 3 3 5 3 0 4 2 4 3 5 5 5 4 0 west	8

Problem H
Nilpotent Matrices
Problem ID: nilpotent
Time Limit: 5

Your algebra professor issued a really mean problem on the latest assignment. "Given a collection of matrices in the monoid $(\mathbb{Z}/2\mathbb{Z})^{d\times d}$ under multiplication, tell me if it possible to generate the 0-matrix with this collection."

Let's put this in more concrete terms. Let $\mathcal{M}$ be a collection of square matrices of the same size $d \times d$ with entries being integers modulo 2 (so they are only 0 or 1).

Say that $\mathcal{M}$ is *nilpotent* if it is possible to choose a sequence of matrices $M_1, M_2, \ldots, M_k$ from $\mathcal{M}$ such that the product $M_1 \cdot M_2 \cdot \ldots \cdot M_k$ is the all-0 matrix. You may use the same matrix of $\mathcal{M}$ more than once in such a sequence.

For example, if $\mathcal{M}$ consists of only the following matrix then it is nilpotent

$$M = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}.$$

This is because $M \cdot M \cdot M = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$.

The following collection is also nilpotent

$$M_1 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}, M_2 = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

For example, the product $M_2 \cdot M_2 \cdot M_1 \cdot M_1 \cdot M_2$ is the zero matrix. Remember that arithmetic is modulo 2.

Input

The first line of input consists of two integers n, d with $1 \leq n \leq 20$ and $1 \leq d \leq 4$. Here, n is the number of matrices in $\mathcal{M}$ and d is the dimension these matrices.

Then n lines follow, each describing one matrix in $\mathcal{M}$. A line describing a matrix contains d^2 values, each value being 0 or 1. The first d values are the first row of the matrix, the second d values are the second row of the matrix, and so on.

Note: The first two samples below correspond to the two examples discussed above in the problem statement.

Output

Output a single line containing the message `nilpotent` or `not nilpotent`, indicating whether the given collection $\mathcal{M}$ is nilpotent or not.

Sample Input	Sample Output
1 3 0 1 1 0 0 1 0 0 0	nilpotent

Sample Input	Sample Output
2 3 1 1 1 1 0 1 0 0 1 1 0 0 1 1 1 1 1 1	nilpotent

Sample Input	Sample Output
2 3 0 1 0 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1	not nilpotent

Problem I
Bob's Bag
Problem ID: bag
Time Limit: 5

Bob was hiking in Jasper national park when his knapsack ripped, scattering all the food he was carrying on the hiking trail. Like any Computing Science student would, Bob immediately took out the scale he always carries around with him on his hikes and started weighing all his food. Bob also read the wrappers to figure out how much nutrition is in each food item.

Help Bob figure out how much food he can carry in his cargo pants before it rips too.

Input

Input begins with two integers: $1 \leq n \leq 20$ indicating the number of food items Bob dropped, and $1 \leq W \leq 10^7$ representing the total weight of food Bob's cargo pants can carry in micrograms.

Next, n lines follow each with two integers: $1 \leq w_i \leq W$ for the weight of the i-th food item in micrographs, and $1 \leq v_i \leq 10^7$ for the value of the i-th food item in micronutrients.

Output

Output a single integer for the maximum amount of micronutrients Bob can carry without ripping his cargo pants (i.e. without exceeding a total weight of W).

Sample Input	Sample Output
3 20 11 20 10 10 10 11	21

This page is intentionally left blank.