Problem A
Getting Started
Problem ID: gettingstarted
Time Limit: 1

1. `on your marks`
2. `get set`
3. `*BANG!*`

Input

Input consists of a single line with a single integer between 1 and 3.

Output

Output a single line with the corresponding message from above. For example, if the input is 2 then output `get set`

The output is case sensitive and must match the message exactly. Do not include any other characters and do not add extra spaces to the end of the line.

You must end the line in a newline chracter `\n`. This is put in by default when using `print()` in Python, but you have to explicitly include it when using Java or C++ (`endl` works with `cout` in C++ as well).

Sample Input	Sample Output
2	get set

Sample Input	Sample Output
3	*BANG!*

Problem B
Octal Equivalence
Problem ID: octal2
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?

If you are unfamiliar with octal numbers, here's a quick tutorial. The usual decimal numbers are based on *powers of 10*. For example, the number 4296 can be written as $4296 = 4000 + 200 + 90 + 6 = 4 \times 10^3 + 2 \times 10^2 + 9 \times 10^1 + 6 \times 10^0$. Note the 10 used as the base of the exponents. In other bases, we simply replace this 10 with the corresponding base. So, in octal, which is base 8, the rightmost digit represents 1s, the next digit represents 8s, the next digits represents 64s, and so on. Then, a number like 6145_8 (where the subscript denotes the base) in octal actually has a value of $6145_8 = 6 \times 8^3 + 1 \times 8^2 + 4 \times 8^1 + 5 \times 8^0 = 6 \times 512 + 1 \times 64 + 4 \times 8 + 5 \times 1 = 3173$ in decimal.

Note, we only use digits between 0 and 7 in octal notation.

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 9 digits and will consist only of digits from 0 to 7.

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
00000012
```
	```
	equivalent
	not equivalent
	not equivalent
	```

Problem C
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 D
Product Inspector
Problem ID: productinspector
Time Limit: 3 seconds

You are given a sequence of slots along a circular conveyor belt, each of which may contain one or more items to inspect. On the conveyer belt are slots labeled from 1 to L, and at the start of each second, the belt rotates so that the slot L moves to where slot 1 used to be, and all other slots i move to where slot $i + 1$ used to be. In other words, if originally the slots are arranged as 123, then after one second, they will be arranged as 312. After one more second, they will be arranged as 231. After one more second, they will return to the original 123.

You are assigned to originally stand in front of slot L at $t = 0$, and to remain there while the belt moves items for you to inspect. It takes one second for you to inspect any item in front of you. Note that a slot could contain multiple items, in which case you can only inspect one of the items before the slot moves past your position. You need to eventually inspect all items! Furthermore, you must inspect items in the given order. How long does it take to inspect all of the items?

For example, in sample input 2 below, you must first wait 65 seconds for the item in slot 35 to reach you, and then inspect it during the one second that it is in front of you. Then, you must wait 88 more seconds for the item in slot 46 to come back around, and then inspect it for one second. Finally, you must wait another 88 seconds for the item in slot 57 to come back around. It will take one final second to inspect this last item, for a total of $65 + 1 + 88 + 1 + 88 + 1 = 244$ seconds.

Input

The first line contains two integers $1 \leq L < 2^{30}$ and $1 \leq n \leq 10^5$, the length of the conveyor belt and the number of products to inspect, respectively. The second line contains n integers a_i, where $1 \leq a_i \leq L$.

Output

Output the amount of time it takes to inspect all of the items in the given order. The answer is guaranteed to be no greater than $2^{31} - 1$.

Sample Input	Sample Output
10 5 10 9 8 7 6	5

Sample Input	Sample Output
100 3 35 46 57	244

Sample Input	Sample Output

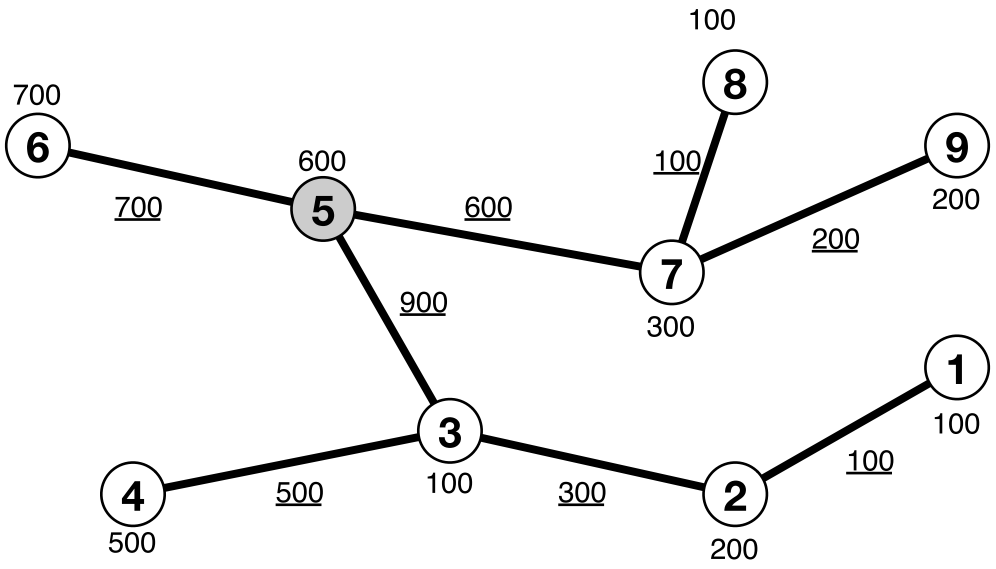
`100 2`
`1 1`
	`200`

Problem E
Stadium
Problem ID: stadium
Time Limit: 8

You just won a government contract to build yet another stadium in Edmonton, but the people of Edmonton have one concern in mind. After an event, thousands of people will leave the stadium at the same time to get back to the neighborhood in which they live. This might cause traffic jams. As responsible city planners, surely you would not build this stadium without first doing a proper study on traffic impact, right? Let's start by finding out how much traffic could be routed on the most congested road.

You have access to good data and can predict how many people from each neighbourhood would attend a typical event. There are n neighborhoods in Edmonton. Since people tend to stick to the major roads, we know that there will be exactly one path between any two neighborhoods. Assume that, after a stadium event, all attendees leave simultaneously, and they all take the direct route back to their neighborhood.

Consider the example depicted below with 9 neighbourhoods (numbered 1 through 9) where the stadium is at neighbourhood 5. The number beside each neighbourhood is the number of attendees who live in that neighbourhood. The underlined number beside each street is then the number of attendees that will travel down that street when driving from the stadium to their neighbourhood. We see that the road from neighbourhood 5 to neighbourhood 3 is used by 900 attendees, and no other road is used by more attendees. So this road has the maximum traffic load after the event.



Input

The first line of input contains two integers n, s, which denote the number of neighborhoods ($2 \leq n \leq 60,000$) and the neighbourhood with the stadium ($1 \leq s \leq n$).

Then n lines follow, each describing a neighborhood. These lines start with the neighborhood number $1 \leq i \leq n$, followed by c_i, the number of neighborhoods connected to neighborhood i. This is then followed by a list of c_i distinct integers (between 1 and n each and all different than i) denoting the neighborhoods connected to neighborhood i. Finally, each line ends with p_i, the number of people in neighbourhood i that will leave the stadium after the event. You are guaranteed $0 \leq p_i \leq 30,000$.

For each neighbourhood, you are also guaranteed there is a unique path from the stadium to that neighbourhood.

Output

Output on a single line indicating the greatest number of people that could end up on any single road.

Sample Input

```
9 5
1 1 2 100
2 2 1 3 200
3 3 2 4 5 100
4 1 3 500
5 3 3 6 7 600
6 1 5 700
7 3 5 8 9 300
8 1 7 200
9 1 7 100
```

Sample Output

```
900
```

Sample Input

```
3 1
1 2 2 3 100
2 1 1 200
3 1 1 300
```

Sample Output

```
300
```

Problem F
Sharing Among Children
Problem ID: sharing
Time Limit: 1 second

The University of Alberta Programmes for Children (UAPC) is a rather unpopular summer camp, contrary to the predictions made by the university's most advanced deep reinforcement learning algorithms. This was overwhelmingly disappointing for the organiser, Ladner. While there was capacity for $50\,000$ participants, only 8 people registered for the camp.

Ladner doesn't let this get him down (too much) and decides to reward the curious minds that did join the camp with cookies...

... a *lot* of cookies.

Unfortunately, given his compromised emotional state, he forgot to check if he ordered an appropriate number of cookies to share evenly among all children. You are here for the free pizza, but in the meantime could you help him out?

Input

The input consists of a single line "`c cookies`" where $0 < c < 10^{10^7}$ is an integer counting the number of cookies Ladner purchased. That is, the integer may have up to 10^7 digits.

Be careful! Casting this to an integer may be too slow...

Output

Output "`Yes!`" (without quotes) if he can distribute the c cookies evenly among the children in the UAPC. Otherwise, output "`This camp has been nothing but a failure.`"

Sample Input	Sample Output
`200 cookies`	`Yes!`

Sample Input	Sample Output
`5 cookies`	`This camp has been nothing but a failure.`

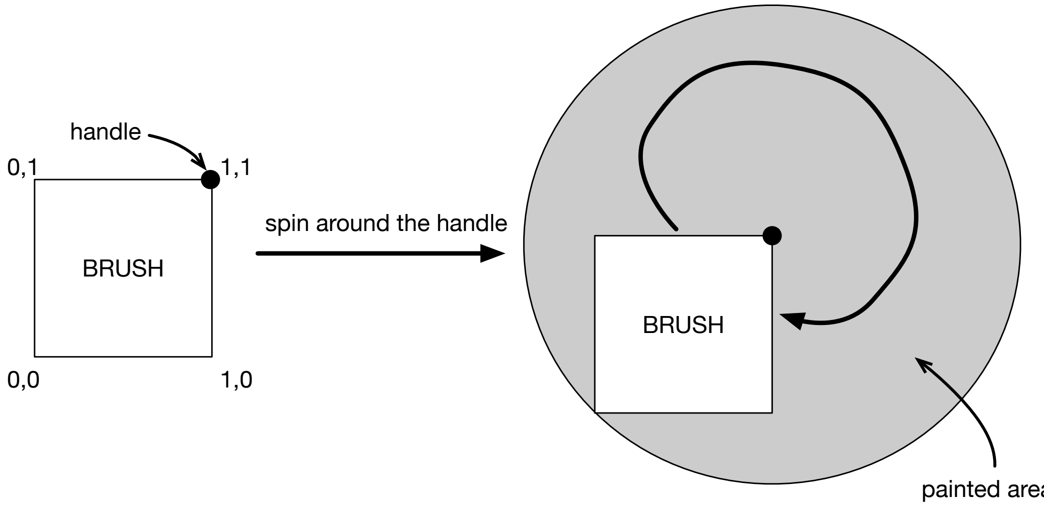
Problem G
Brush Pivot
Problem ID: brush
Time Limit: 3 second

In the land of Uncannily Asymmetric Polygonal Creations (UAPC), Savitch realised that all of the walls of his house are the same colour. This simply cannot do, so he pulls out his trusty paintbrush from space. His paintbrush is actually quite famous for its ability to cover a lot of space, and people in UAPC like to call it PSPACE. Savitch prefers it be called NPSPACE because of how nifty it is, but there doesn't seem to be much use in making the distinction—it's the same thing, after all.

Anyway, some carelessness resulted in the brush being reduced to handlelessness. Instead of fretting over this, Savitch sees this as an opportunity to improve its space bound. He wants to attach a handle to PSPACE so that when pivoting about this handle, he can paint the maximum area. In other words, Savitch's brush is a two-dimensional polygon that is placed flat on the wall to be painted, and a pivot point is chosen where the handle will be attached. By rotating the entire brush around this pivot point, some area of the wall will be covered by the brush. Nifty, isn't it?

Savitch is a fairly independent guy and already figured out where to put the handle optimally. Given optimal placement of the handle, how much area can he paint with (N)PSPACE after a single full spin?

Below is an illustration of the first sample input and one possible placement of the handle on the brush/polygon to maximize the resulting painted area if the brush is spun about the handle.



Input

The first line contains an integer $3 \leq p \leq 1000$ denoting the number of vertices on Savitch's brush.

The next p lines contain two space-separated real numbers $-1000 \leq x_i, y_i \leq 1000$ indicating that the ith vertex is at position (x_i, y_i). The coordinates are given in centimetres.

Output

Output the maximum area (in cm^2) that PSPACE can cover, rounded down to exactly five decimal points.

Sample Input	Sample Output

4
0.00000 0.00000
1.00000 0.00000
1.00000 1.00000
0.00000 1.00000
	6.28319

Sample Input	Sample Output

4
0.00000 5.00000
1.00000 0.00000
0.00000 -5.00000
2.00000 0.00000
	314.15927

Problem H
Lockpick
Problem ID: lockpick
Time Limit: 3 second

Congratulations! You are the 49992nd student to register for Unlawful Access to Prohibited Containers (UAPC) summer school! Your prize is a locked box labeled "Proof that **P = NP**". This proof is worth US$1,000,000 and probably also a lot of cryptocurrency!

As a highly skilled programmer you are inherently very lazy so you're dreading having to present the proof in order to claim your US$1,000,000. Why couldn't they just put the money in the box??

Anyways, before you can even claim your prize, you need to answer a skill testing question. If you can't solve it in polynomial time, then clearly you won't understand the proof anyway, so Ladner will transfer you to the University of Alberta Programmes for Children (UAPC) summer camp and give the box to the 49991st student instead. If this is the case then unfortunately you are ineligible to recieve cookies as part of the summer camp because the registration deadline has already passed.

The lock on the box has n pins. All pins start in the 0 position, and when you "set" a pin, it goes to the 1 position. For the lock to open, all pins must be in the 1 position. Additionally, because you don't have the key, you need to set the pins in the correct order.

The fun of it is that you don't know what order to set the pins in, so you'll have to keep trying different orders until you find the correct one. If at any point you set a pin out of order, the lock will jam and you'll have to shake it vigorously to reset all the pins back to the 0 position before you can try again.

Your skill testing question is this: given n, what is the expected number of times that the lock will jam before you get it open, assuming you remember your past mistakes?

Explanation for First Sample:
You have a 50/50 chance of setting the correct pin at the start. If you do, then you will, of course, set the second pin correctly and open the box.

If you do not, the lock jams and you have to restart. But when you restart, you at least remember which pin should not be set first. So you start by setting the other pin, and then open the box by setting the last pin.

With probability 0.5 there are no jams. With probability 0.5 there is one jam. So the expected number of jams is $0.5 \cdot 0 + 0.5 \cdot 1 = 0.5$.

Input

A single integer n, where $1 \leq n \leq 10000$.

Output

Print the expected number of times that a lock with n pins will jam before you can open it, specified to exactly one decimal point.

Sample Input	Sample Output
`2`	`0.5`

Sample Input	Sample Output
`5`	`5.0`

Problem I
Ugh! Albert is a Picky Child
Problem ID: pickychild

Time Limit: 3 second

It looks like somehow you've been entrusted with taking care of four-year old Albert for a few hours. At first, it seemed like a daunting task, but then you remembered what you did to pass the time at that age: looking at long lists of integers! You figure that babysitting will be easy if you can simply print out a list of consecutive integers and let Albert stare at it.

Unfortunately, you have just been told that Albert is a somewhat picky child. He only likes integers that have a repeated digit in two consecutive positions. What a strange child! For example, Albert likes the integers 33, 77777, and 12344, as there exist consecutive 3s, 7s, and 4s, respectively. Albert does not like the integers 8 or 49494.

You could print out the integers from some m to some n inclusive, but would it be worth your time? How many integers would Albert even like? Perhaps you can write a program to help answer this question.

Input

Input consists of one line containing two integers m and n with $1 \leq m \leq n \leq 100,000$, indicating that you will print all integers from m to n inclusive (*i.e.*, m, $m+1$, ... , n).

Output

Output the number of integers that Albert would like in the specified range.

Sample Input	Sample Output
`10 23`	`2`

Sample Input	Sample Output
`999 999`	`1`

This page is intentionally left blank.

Problem J
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`