Problem A. Archivist

Time limit:	2 seconds
Memory limit:	512 megabytes

The team of problemsetters of Northwestern Russia Regional Contest welcomes you! Our regional contest was founded in 1995 under the name “Collegiate Programming Championship of St Petersburg”. Here is the list of the contest winners:

- 1995: ITMO
- 1996: SPbSU
- 1997: SPbSU
- 1998: ITMO
- 1999: ITMO
- 2000: SPbSU
- 2001: ITMO
- 2002: ITMO
- 2003: ITMO
- 2004: ITMO
- 2005: ITMO
- 2006: PetrSU, ITMO
- 2007: SPbSU
- 2008: SPbSU
- 2009: ITMO
- 2010: ITMO
- 2011: ITMO
- 2012: ITMO
- 2013: SPbSU
- 2014: ITMO
- 2015: ITMO
- 2016: ITMO
- 2017: ITMO
- 2018: SPbSU
- 2019: ITMO

Help the contest archivist to remember the results of each contest and write a program that will read the year and print contest winners of that year in exactly the same format as above.

Input

The only line of input contains a single integer y ($1995 \le y \le 2019$), denoting the year. You don’t need to process year numbers less than 1995 or greater than 2019, or incorrect year formats. It is guaranteed that you will be given a number between 1995 and 2019, inclusive.

Output

Print the winner of the contest in year y exactly in the same format as in the list above.

Examples

standard input	standard output
`2006`	`PetrSU, ITMO`
`2019`	`ITMO`
`2018`	`SPbSU`
`2003`	`ITMO`

Problem B. Bicycle

Time limit:	2 seconds
Memory limit:	512 megabytes

After a long time at home during the quarantine, in November you decided to go to work by bicycle! Since you do not have your own bicycle, you have to rent one. The bike rental allows you to choose one of two monthly options:

- The monthly fee is a roubles. Every day, the first 30 minutes are free, and every minute above that costs x roubles.
- The monthly fee is b roubles. Every day, the first 45 minutes are free, and every minute above that costs y roubles.

There are 21 working days in November, and you spend T minutes commuting to work and back home *in total* every day. Calculate how many roubles you would spend if you use either one of two monthly options.

Input

The first four lines of the input contain integers a, x, b, and y ($0 \le a, x, b, y \le 100$), each on a separate line. The last line contains a single integer T ($1 \le T \le 1440$) — the total time spent on a bicycle during each day.

Output

The only line of the output should contain two integers — the amount of money you would spend on the first option and the second option, respectively.

Examples

standard input	standard output
10 1 20 5 50	430 545
10 10 10 10 42	2530 10
10 10 10 10 27	10 10

Problem C. Corrupted Sort

Time limit:	2 seconds
Memory limit:	512 megabytes

This is an interactive problem.

Chloe wants to test her sorting skills. She has n cards with distinct integers from 1 to n written on them. She asks her little brother Connor to first blindfold her and then arrange all cards in a row in some order. Positions of cards are numbered from 1 to n from left to right.

Chloe doesn't know the order of the cards, but she wants to sort them, so that the leftmost card has number 1 and the rightmost card has number n on it. Formally, for each i she wants the card on position i to have number i on it. To achieve the goal, Chloe can ask Connor to do one or more operations.

Each operation can be denoted by two integers pos_i and pos_j ($1 \le pos_i < pos_j \le n$). Connor looks at the cards on positions pos_i and pos_j, and if the card on position pos_i has a bigger number than the card on position pos_j, he swaps them. Otherwise, he does nothing. Connor also tells Chloe if he swapped the cards or not.

To make the game more interesting, after every $2n$ operations Connor chooses two distinct cards uniformly at random and swaps them without telling anything to Chloe.

If after some of Chloe's operations all cards become sorted, Chloe wins. Help Chloe to sort all cards using at most 10 000 operations.

Interaction Protocol

Your program should start by reading a single integer n ($2 \le n \le 50$) — the number of cards.

Your program can ask to do the specified operation one or more times. To do the operation, your program should print its description formatted as "pos_i pos_j" — two distinct numbers denoting the positions of cards ($1 \le pos_i < pos_j \le n$). Remember to flush the output after printing each operation description. Then your program should read the outcome of the operation — it will be a single word `SWAPPED` if the cards were swapped, or a single word `STAYED` if nothing changed.

If instead of an operation outcome your program receives a single word `WIN`, it means that the cards have become sorted. Your program should terminate silently after that, it is not allowed to output any extra operations.

After every $2n$ operations two distinct cards are chosen randomly and swapped without telling anything to your program. Each pair of cards has equal probability to be chosen. This swap is not counted as an operation.

Example

standard input	standard output
3	
	1 2
SWAPPED	
	1 3
STAYED	
	2 3
WIN	

Note

Initial card ordering in the example was 3 1 2 (that is, the card on position 1 had number 3 on it, the card on position 2 had number 1, and the card on position 3 had number 2).

Problem D. Display

Time limit:	2 seconds
Memory limit:	512 megabytes

St. Petersburg Supway tests new SupLED ticker displays. The display shows a monochrome sliding text line printed in SupFont. The text is only shown once and does not loop. Each SupFont character has a fixed size of $w \times h$ pixels, where each pixel is either *on* or *off*. There is a column of *off* pixels between consecutive characters of the text. The text sliding speed is one pixel per tick.

The main service life factor is the number of switches: when a pixel turns from *on* to *off* or vice versa. A pixel breaks after s switches. If a pixel's state is not changed between two ticks, no switches occur.

The services team needs to estimate the shortest text that causes any of the pixels to break. All pixels are *off* before the text is shown.

Input

The first line of the input contains four integers n, w, h, and s — the number of characters in SupFont, width and height of each character, and the number of switches after which a pixel becomes broken ($1 \le n \le 94$; $1 \le w, h \le 30$; $1 \le s \le 10^6$).

The following lines contain SupFont character descriptions. The first line of each description contains an ASCII character (only characters with codes between 33 and 126, inclusive, are used). The character's image with h lines of length w follows, '`#`' denotes an *on* pixel, and '`.`' denotes an *off* pixel. Each image has at least one *on* pixel.

All described ASCII characters are distinct, but some of them may have the same image.

Output

Output a single line containing at most s characters: the shortest text that, when shown on a display, causes some pixel to switch at least s times. If there are multiple solutions, output any of them.

Example

standard input	standard output
3 4 5 16 I .#.. #.#. .#.. .#.. ###. C .##. #..# #... #..# .##. P ###. #..# ###. #... #...	ICPC

Problem E. Easy Compare-and-Set

Time limit:	2 seconds
Memory limit:	512 megabytes

Let us define "Compare-and-Set" operation for a global variable v. The operation checks if the variable is equal to a. If that's true, the variable value changes to b and the operation succeeds. Otherwise, the variable doesn't change and the operation fails. Let us denote the operation as $\mathrm{CAS}(a, b)$.

Imagine that you are given a list of such operations $\mathrm{CAS}(a_1, b_1), \ldots, \mathrm{CAS}(a_n, b_n)$. Also, you are given an initial value for the variable, c, and a list of wishes $w_1, \ldots w_n$, where w_i tells whether the operation $\mathrm{CAS}(a_i, b_i)$ should be successful. Your task is to determine the order of operations execution so that all the wishes are satisfied.

Input

The first line contains two integers n and c ($1 \le n \le 10^5$; $1 \le c \le 10^9$) — the number of operations and the initial value of the variable.

Each of the next n lines contains three integers a_i, b_i, w_i ($1 \le a_i, b_i \le 10^9$; $0 \le w_i \le 1$), denoting $\mathrm{CAS}(a_i, b_i)$ operation that you wish to be successful if $w_i = 1$ and unsuccessful if $w_i = 0$. The operations are numbered from 1 to n in order of input.

Output

If no correct order of operations exists, output a single word "`No`".

Otherwise, output a word "`Yes`" followed by n distinct integers $p_1, p_2, \ldots p_n$ ($1 \le p_i \le n$), meaning that operation p_1 should be executed first, then operation p_2, and so on. If there are several possible orders, output any of them.

Examples

standard input	standard output

4 1
1 2 0
1 2 1
2 3 1
3 4 0
	Yes
	4 2 1 3

3 1
1 2 1
1 2 1
1 2 0
	No

Problem F. Futures Market Trends

Time limit:	2 seconds
Memory limit:	512 megabytes

Fedor wants to earn some extra money out of oil futures. His analysis is based on *trends* and historical data of d previous days.

A cost sequence of at least three distinct days $c_0, c_1, c_2, \cdots, c_n$ forms a *trend of power P* if average cost change is at least P times bigger than the standard deviation of cost changes.

Formally, average cost change is $A = \frac{1}{n}\sum_{i=1}^{n}(c_i - c_{i-1})$. The standard deviation of cost changes is $D = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(c_i - c_{i-1} - A)^2}$. The sequence forms a *positive trend of power P* if $\frac{A}{D} \geq P$, and a *negative trend of power P* if $\frac{A}{D} \leq -P$.

We assume that if $A > 0$ and $D = 0$ then a positive trend of any power is formed. Similarly, if $A < 0$ and $D = 0$ then a negative trend of any power is formed. If $A = 0$ then no trend is formed, even if $D = 0$.

To check his theory, Fedor needs to count the number of subsegments of historical data which form positive and negative trends of the given power. Can you help him?

Input

The first line contains an integer d $(3 \leq d \leq 3\,000)$ and a real number P $(0.001 \leq P \leq 1000)$ — the number of days in Fedor's historical data and the required trend power. P is given with at most 9 digits after the decimal point.

The next line contains d integers $c_1, c_2, \ldots, c_d$ $(-1000 \leq c_i \leq 1000)$. For each i, c_i is the cost of oil futures at the end of the i-th day.

Output

Print two integers — the number of subsegments of historical data forming a positive and a negative trend of power P, respectively. It's guaranteed that the answer would not change if one changes P by no more than 10^{-8}.

Examples

standard input	standard output

`6 0.2`
`100 110 120 30 40 50`
	`2 8`

`6 0.7`
`100 110 120 30 40 50`
	`2 2`

`10 1.234`
`236 250 227 224 201 198 198 182 -376 100`
	`0 5`

Note

The third example test case demonstrates the real prices of oil futures in April 2020 (in dollars, multiplied by 10 and rounded). Be careful with theories like the one Fedor made.

Problem G. Grammar Path

Time limit:	2 seconds
Memory limit:	512 megabytes

You are given a context-free formal grammar in Chomsky reduced form (see Notes section for an explanation of these terms), and a directed graph with each edge labeled with a terminal of the grammar.

Find the length of the shortest path in the graph from vertex s to vertex t such that concatenation of the labels on this path is in the language of the grammar, or state that there is no such path.

Input

The first line contains the number of productions in the grammar p ($1 \le p \le 100$).

Each of the next p lines contains a production in the form of either '`A -> BC`' or '`A -> a`'. The lowercase English letters are terminals, the uppercase English letters are nonterminals, and the uppercase letter '`S`' is the starting nonterminal. It's guaranteed that `S` appears as left hand side of at least one production.

The next line contains four integers n, m, s, and t ($1 \le s, t \le n \le 26$; $0 \le m \le n^2$), denoting the number of vertices in the graph, the number of edges in the graph, and indices of the start and the finish.

Each of the next m lines contains a description of an edge in the form '`u v x`', denoting an edge from vertex u to vertex v labeled with x ($1 \le u, v \le n$; x is a lowercase English letter). There are no multiedges in the graph, but there might be loops and different edges from u to v and from v to u.

Output

If there is no path from s to t with labels forming a string from the language, output '`NO`'. Otherwise, output the length of the shortest such path.

Note

Informally speaking, a context-free formal grammar is a set of terminals (lowercase English letters in this problem), a set of nonterminals (uppercase English letters in this problem), and a set of rules of how a nonterminal can be replaced with a string of nonterminals or terminals.

Chomsky reduced form is a form where each rule is a replacement with either a single terminal or exactly two nonterminals. In fact, any context-free grammar which does not generate the empty string can be converted to Chomsky reduced form.

A string of terminals is in the language of the grammar if one can use the rules to convert a string of a single starting nonterminal into the given string. You can get some formal details at `https://en.wikipedia.org/wiki/Context-free_grammar`.

The formal grammar in the two last examples defines all non-empty correct bracket sequences, with opening brackets denoted with '`c`' and closing brackets denoted with '`j`'.

Examples

standard input	standard output

```
5
S -> AB
A -> a
A -> AA
B -> BB
B -> b
8 8 1 4
1 2 a
2 3 b
3 4 a
1 5 a
5 6 a
6 7 a
7 8 b
8 4 b
```
	`5`

```
6
S -> SS
S -> LA
S -> LR
A -> SR
L -> c
R -> j
4 5 1 1
1 2 c
2 3 c
3 1 j
1 4 j
4 3 j
```
	`12`

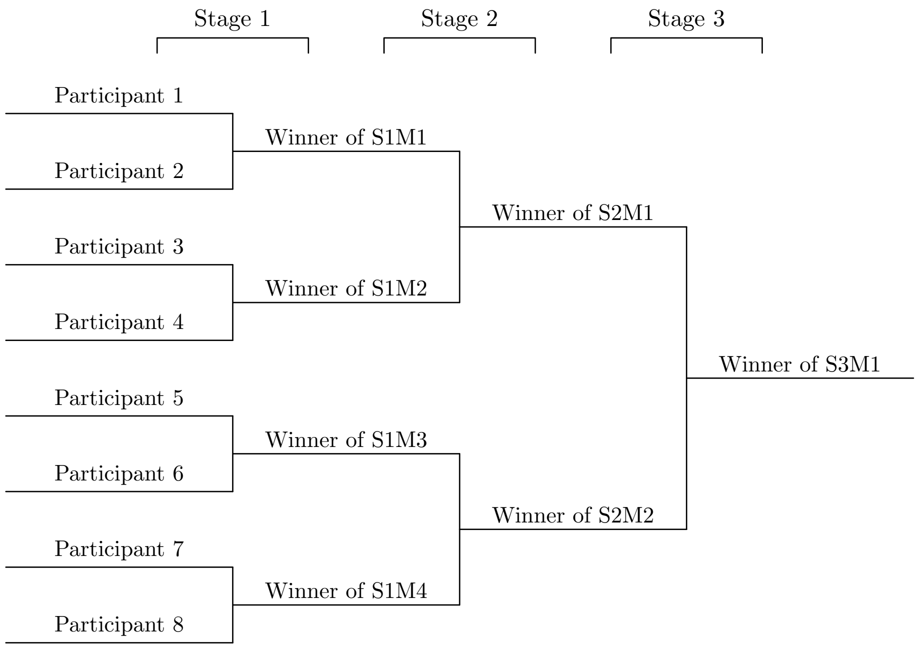
```
6
S -> SS
S -> LA
S -> LR
A -> SR
L -> c
R -> j
4 5 1 4
1 2 c
2 1 c
2 3 c
3 4 j
4 3 j
```
	`NO`

Problem H. Heroes of Coin Flipping

Time limit:	2 seconds
Memory limit:	512 megabytes

It is year 2030. Professional coin flipping has become the most popular game on the Internet. Latest coin flipping World Championship gathered the record amount of participants from around the world! There are 2^k participants, competing in a single-elimination tournament for the title of the Coin Flipping World Champion. For simplicity, we will refer to participants as numbers from 1 to 2^k.

In the first stage, participants are split into pairs: 1 against 2, 3 against 4, ... $2^k - 1$ against 2^k. The winner of each match advances to the next stage. Every next stage goes on similarly: remaining participants are sorted by their number, then split into pairs to play their matches. The last k-th stage is just one match that crowns the world's best coin flipper. All participants' coin-flipping skills are almost identical to each other: in a match with any two participants, the probability of one of them winning is exactly $\frac{1}{2}$.



An example of a single-elimination tournament with 8 participants. SXMY is a shorthand for the Y-th match of the X-th stage.

Hedy missed the live broadcast of the tournament, so she's going to watch all the recordings after the fact. She already knows the list of all participants (and their numbers), but doesn't know any match outcomes. Friends suggested Hedy n matches to watch first (without spoiling the result, of course!). First she is going to watch these n matches in a specific order, then watch all the remaining matches in random order.

Hedy considers a match *exciting* if she doesn't know the winner of that match before watching it. For example, after watching the final, both semifinals will not be exciting for her, because she will have already seen the semifinals winners competing in the final.

Find the expected number of exciting matches, averaged over all possible tournament outcomes and watch orders.

Input

The first line of the input contains two integers k and n ($1 \le k \le 30$; $0 \le n \le \min(2^k - 1, 10^5)$). Next n lines describe the suggested matches in order Hedy is going to watch them. Each line contains two integers s_i and m_i, the stage and the number of the match within that stage ($1 \le s_i \le k$; $1 \le m_i \le 2^{k-s_i}$; all pairs (s_i, m_i) are distinct).

Output

Print a single real number — the expected number of exciting matches. The absolute or relative error should not exceed 10^{-9}.

Examples

standard input	standard output

2 3
1 1
2 1
1 2
	2.0

2 1 1 1	2.5

3 2
3 1
1 4
	2.25

4 0	6.833333333333333

Note

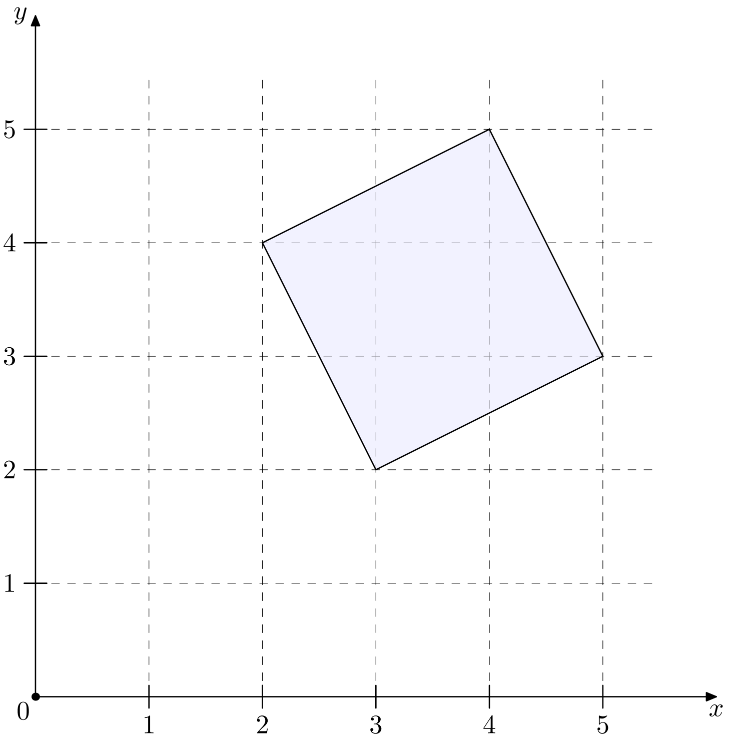
In the first example, there is no randomness. The first two matches are always exciting, the third one is always not.

In the second example, the first match is exciting. There are two possible orders. In case the semifinal match comes before the final match, both matches are exciting. In the other case, the semifinal will not be exciting as Hedy watches it after the final. The expected value is $1 + \frac{1}{2}(2 + 1) = 2.5$.

Problem I. Integer Square

Time limit:	2 seconds
Memory limit:	512 megabytes

Iris likes to draw different shapes on a grid paper. Her favorite shape is a square. She has recently tried to draw a square with area s, but since she uses a grid paper, she wants all the vertices of the square to have integer coordinates.



Can you help her find such a square?

Input

The input contains a single integer s ($1 \le s \le 1000$).

Output

If it is possible to construct the required square, output four pairs of integers: the coordinates of the vertices of the square, in any order. All coordinates should be in the range from -10^9 to 10^9. If there are multiple answers, output any of them.

If it is impossible to construct the required square, print "`Impossible`".

Examples

standard input	standard output

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

`3`	`Impossible`

Problem J. Joint Password Storage

Time limit:	2 seconds
Memory limit:	512 megabytes

Johnny is the developer of Joint Password Storage (JPS). Storing passwords in plaintext is a bad idea. JPS splits each password into several parts and stores them separately.

A password is a string consisting of digits and English letters. Bitwise XOR of all parts should be equal to the password. To attract less attention, Johnny decided that each part should look like something ordinary, like an arithmetic equality. What could be more ordinary than things like "$2 + 2 = 4$"?

Formally, a valid split of a password is a set of correct arithmetic equalities of the same length as the password, for which bitwise XOR of ASCII codes of characters on each position is equal to the ASCII code of the corresponding password character. A correct arithmetic equality is a string described by the following grammar, with both expressions evaluating to the same value. Operator precedence is standard: expressions in brackets of any type take precedence over all other operations, multiplication takes precedence over addition and subtraction, and operators with the same level of precedence are evaluated from left to right.

```
    ⟨equality⟩  ::=  ⟨expression⟩ '=' ⟨expression⟩
  ⟨expression⟩  ::=  ⟨term⟩ | ⟨expression⟩ '+' ⟨term⟩ | ⟨expression⟩ '-' ⟨term⟩
        ⟨term⟩  ::=  ⟨multiplier⟩ | ⟨term⟩ '*' ⟨multiplier⟩
  ⟨multiplier⟩  ::=  ⟨number⟩ | '(' ⟨expression⟩ ')' | '[' ⟨expression⟩ ']' | '{' ⟨expression⟩ '}'
      ⟨number⟩  ::=  '0' | ( '1' | ... | '9' ) ( '0' | ... | '9' )*
```

For your convenience, ASCII codes of all related characters are provided below:

(	)	*	+	-	0-9	=	A-Z	[	]	a-z	{	}
40	41	42	43	45	48-57	61	65-90	91	93	97-122	123	125

Your task is to write a splitting module which converts a password into bitwise XOR of several correct arithmetic equalities.

Input

The first line contains a single integer P ($1 \le P \le 50$) — the number of passwords to split. Each of the next P lines contains a single string — password s ($10 \le |s| \le 50$). Passwords can contain digits and both lowercase and uppercase English letters.

Output

For each password, if there is no valid split, output a line with a single word "`NO`".

Otherwise, the first line should contain a single word "`YES`". The next line should contain one integer k ($1 \le k \le 1000$) — the number of equalities in your split. Each of the next k lines should contain one equality. It can be proven that if a solution exists, there is a solution where k doesn't exceed 1000.

Example

standard input

```
3
1915090454
CanIAlwaysSplitIt
2020NorthwesternRussiaRegionalContestTaskJ
```

standard output

```
YES
3
9+91*9=828
1+19+0=4*5
999=1008-9
NO
YES
7
420+[1*2*3*4*5*6]=140+[1*10*1*[20-5-5]*10]
10-1-{1}-{1}-{1}-{1}={1}+{1}+{1}+{{1}+{1}}
739=1+{3}*{2}*{9}*{9}-3+{2}*7*11*1+{100}+1
602211592866240={54321*67890}*9*{8*7*6*54}
51-188*1*0*600198090+5=[0]*(1039-25-4)+8*7
0*990-5127*11590*740*0*[3]*90=8*0*4885*2*8
3*0-0*818*39=0*(6+10*(64))*200*(93+6+8+19)
```

Problem K. Keys and Locks Boolean Logic

Time limit:	2 seconds
Memory limit:	512 megabytes

Karen and her friends play in a music band in Karen's garage. She has complicated thoughts on which subsets of the band should be allowed to enter the garage. She designed a boolean formula F in which each variable denotes the presence of a certain band member. She wants a group of people to be able to enter the garage if and only if F is true.

You are asked to create a system of wires and locks that will fulfill the formula F. Each band member will receive a key that opens all the locks that are denoted by their letter.

For example, suppose that $F = a \vee (b \wedge c)$ and band members a and b are present. Then they should be able to enter the garage with their keys because $true \vee (true \wedge false) = true$.

A band member doesn't have to use their key to open all the locks possible. For example, suppose that $F = a \oplus b$. Then band member a alone should be able to enter the garage because $true \oplus false = true$. But if both a and b come to the garage, they can just disregard b's key and open the garage using a's one. However, $true \oplus true = false$, so this formula F is not fulfillable.

The system must be a rectangular grid of size at most 50×50 containing horizontal wires '`-`', vertical wires '`|`', wire connections '`+`', and locks. The upper leftmost and the upper rightmost cells of the grid must contain wire connections that will be attached to the doors of the garage. The garage doors stay closed as long as there is a path between these two connections via wires and locked locks.

You are to design such a system corresponding to the given formula F, or to state that there is no such system.

Input

The input contains a non-empty boolean formula F.

The formula can contain letters `a`, ..., `h` denoting the presence of band members, operators "`and`", "`or`", "`not`", and parentheses. The length of the formula does not exceed 2020. The "`not`" operator has the highest priority. The "`and`" operator has higher priority than the "`or`" operator. There are single spaces around each "`and`", around each "`or`", and after each "`not`". There are no spaces other than those.

Output

If it is impossible to design the desired system, print one line "`IMPOSSIBLE`".

Otherwise, output a rectangular grid of characters — the system corresponding to formula F. The system can only contain spaces, '`-`', '`|`', '`+`', and letters corresponding to the band members that are mentioned in the input.

The upper leftmost and the upper rightmost cells should be wire connections. The width of the grid should be between 2 and 50, inclusive, the height of the grid should be between 1 and 50, inclusive.

Character '`-`' must be used for a wire if and only if it connects something on the left and something on the right with empty cells below and above. Similarly, '`|`' must be used for a wire if and only if it connects something below and something above with empty cells on the left and on the right.

Examples

standard input	standard output

`a or (b and c)`	`+-+ +b-+`
	`| | |  |`
	`+-a-+-c+`

`(a or f) and ((a and g) or (a and h))`	`+a-f+`
	`|   |`
	`+g+h+`
	`a | a`
	`+-+-+`

`a and not b or not a and b`	`IMPOSSIBLE`
`b or not b`	`+ +`

`d and not d`	`+  +---+  +---+--+---+`
	`|  |      |   |  |`
	`+--+      +---+  |`
	`|  |      |      |`
	`+  +------+      +---+`

Problem L. Lost Permutation

Time limit:	2 seconds
Memory limit:	512 megabytes

This is an interactive problem.

You once had a permutation π of size n. And now it's gone. All you have left is an old device you made while studying group theory. To try and recover π you can input a permutation f of size n into this device. This device will then display a permutation $\pi^{-1} \circ f \circ \pi$. Find π using at most two interactions with the device.

A permutation of size n is a sequence of n distinct integers from 1 to n. The *composition* of two permutations a and b is a permutation $a \circ b$ such that $(a \circ b)_i = b_{a_i}$. That is, if we consider a permutation as an action on n elements, moving element at position i to a_i, then $a \circ b$ is the action that applies a, then applies b, so that element at position i first moves to a_i, then moves to b_{a_i}. Note that some definitions of composition use the reverse order.

The inverse permutation π^{-1} is a permutation σ such that $\sigma_{\pi_i} = i$. The composition of a permutation and its inverse is equal to an identity permutation: $(\pi \circ \pi^{-1})_i = (\pi^{-1} \circ \pi)_i = i$ for all i from 1 to n. For example, if $a = (4, 1, 3, 2)$ and $b = (3, 2, 1, 4)$, then $a \circ b = (4, 3, 1, 2)$, $a^{-1} = (2, 4, 3, 1)$ and $a^{-1} \circ b \circ a = (1, 2, 4, 3)$.

Interaction Protocol

Your program has to process multiple test cases in a single run. First, the testing system writes t, the number of test cases ($t \geq 1$). Then, t test cases should be processed one by one.

In each test case your program should start by reading the integer n ($3 \leq n \leq 10^4$), the size of permutation π. The sum of n over all test cases does not exceed 10^4. Then, your program can make queries of two types:

- `?` $f_1\, f_2\, \ldots\, f_n$, values $f_1, f_2, \ldots, f_n$ form a permutation of $1, 2, \ldots, n$. The testing system responds with a permutation $g_1, g_2, \ldots, g_n$, where $g = \pi^{-1} \circ f \circ \pi$.
- `!` $\pi_1\, \pi_2\, \ldots\, \pi_n$ — your guess for the secret permutation.

You can use at most two queries of the first type in each test case. After your program makes a query of the second type, it should continue to the next test case (or exit if that test case was the last one).

Example

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

Note

There are two test cases in the first test. In the first test case, $\pi = (4, 1, 3, 2)$ is the only permutation that satisfies $\pi^{-1} \circ (3, 2, 1, 4) \circ \pi = (1, 2, 4, 3)$ and $\pi^{-1} \circ (2, 4, 3, 1) \circ \pi = (2, 4, 3, 1)$. In the second test case, based on the interaction, π can be equal to either $(1, 3, 2)$, $(2, 1, 3)$, or $(3, 2, 1)$. The solution got lucky and guessed the correct one: $(3, 2, 1)$.

Problem M. Mind the Gap

Time limit:	2 seconds
Memory limit:	512 megabytes

Mika is playing the Mind game with her friends.

The game is played with cards, a single integer value is written on each card. All values written on the cards are distinct. Players keep playing cards, building a single pile of cards on the table. Initially the pile contains a single card with integer 0 on it. Each player is given a single card with a value from 1 to 10^9. Then players start playing the cards in any order. When a player plays a card, they place it on the top of the pile. The goal is to play all the cards in such a way that the card values increase from the bottom to the top. If any player didn't play their card or the pile is not increasing, the players lose, otherwise they win. No communication between players is allowed after the cards are distributed.

Mika and her friends came up with a strategy. They agree on a single integer d before the game. During the game, if a player's card value is x, and the top value of the pile is y, such that $x - y \le d$, then the player plays their card. If $x - y > d$, the player doesn't play their card. If several players play their cards at the same time, these cards might be placed on the top of the pile in any order, and this order is not controlled by the players.

You are given the card values that are going to be distributed to the players. Your task is to find an integer d for the players' strategy that guarantees them a win.

Input

The first line contains an integer n — the number of players playing the Mind game ($3 \le n \le 100\,000$).

The second line contains n integers — the card values distributed to the players.

All given card values are distinct, positive and don't exceed 10^9.

Output

Print a single integer d that Mika and her friends should use to guarantee a win in the game using their strategy. If no such d exists, print 0. If several values of d exist, print any of them.

Examples

standard input	standard output
5 5 13 2 10 8	4
5 4 13 2 8 7	0

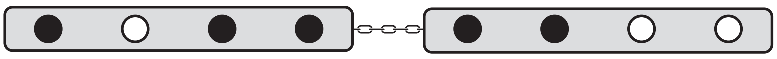
Note

In the first example $d = 3$ would also be a correct answer.

Problem N. Nunchucks Shop

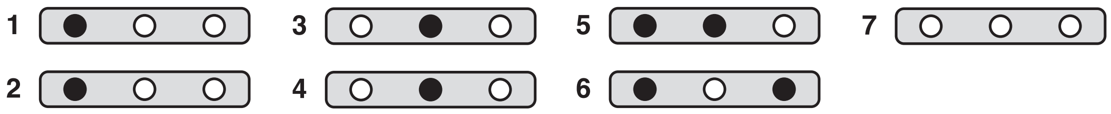
Time limit:	2 seconds
Memory limit:	512 megabytes

Nathan owns a shop that sells souvenir nunchucks with unique design. Nunchucks is a traditional martial arts weapon consisting of two sticks connected with a chain. In Nathan's design, each stick is inlayed with n gemstones arranged in a row. These gemstones are either quartz or onyx, and they create a nice black-and-white pattern. For aesthetics reasons, Nathan sells only nunchucks with exactly k onyxes on both sticks in total. For example, here is one of the possible designs for $n = 4$ and $k = 5$.

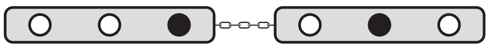


Recently Nathan decided that it will be good to be able to sell nunchucks with every possible design. But that will require him to have nunchucks with all possible designs in the storage, and the number of possible designs is huge!

Thus, Nathan decided to compromise. He will have a number of sticks in his storage. When a customer orders some design, Nathan will take two sticks from the storage and combine them with a chain. Sticks are symmetrical, and Nathan can connect the chain to any end of the stick. For example, if $n = 3$ and $k = 2$, and Nathan has these sticks in the storage:



Then he can make nunchucks of any possible design. For example, if the customer asks for nunchucks of this design:



Then Nathan can make it from sticks 1 and 3.

Now Nathan wonders: what is the minimum number of sticks he should have in the storage to be able to make nunchucks of any possible design? Help him find this number.

Input

The input contains two integers n and k ($1 \le n \le 50$; $0 \le k \le 2 \cdot n$).

Output

Output one integer: the minimum number of sticks Nathan should have in the storage.

Examples

standard input	standard output
3 2	7
4 1	3
5 0	2
50 50	626155273417404