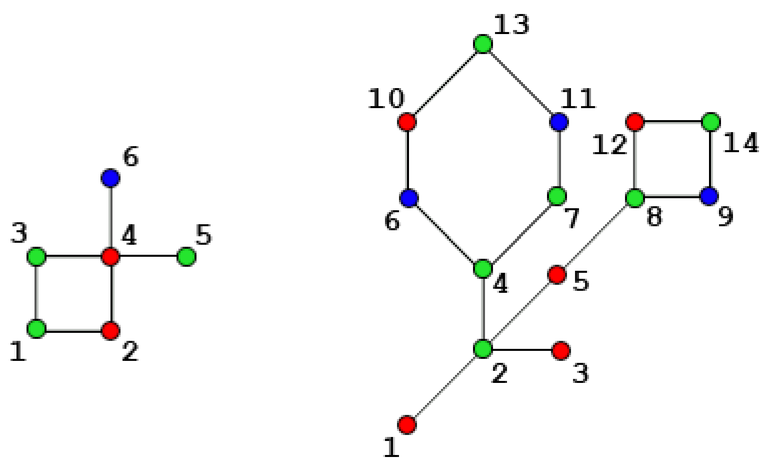
Problem A. Alone in the Cactus

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

Cactus is a connected undirected graph in which every edge belongs to at most one simple cycle.

You are given a *cactus* having n vertices and m edges. Each vertex of this *cactus* is either red, green or blue. All the vertices are numbered with sequential positive integers from 1 to n. The examples of such graphs are shown in the figure:



Initially, you are standing at the vertex s. Then you start to move to some adjacent vertex, which hasn't been visited before, until no such vertex exists. The choice of each eligible vertex is equiprobable. When there is no such vertex, you are stopped at some vertex t. If t is colored red or green — the process is stopped. If t is colored blue — the process should be restarted from the vertex s again.

You are to write the program that for given *cactus* computes the probability $\frac{p}{q}$ that the process will be stopped when you are standing at a red vertex and outputs integer $p \cdot q^{-1} \mod 1000000007$ $(10^9 + 7)$. Here q^{-1} is a modular multiplicative inverse of an integer q modulo 1000000007.

Input

The first line of input contains three space-separated integers n, m and s ($2 \le n \le 10^5$, $m \ge n - 1$, $1 \le s \le n$). The second line of input contains n characters, i-th of them denotes the color of vertex i: '`R`' denotes red, '`G`' — green and '`B`' — blue. Each of the following m lines contains two integers u_i, v_i — the numbers of vertices connected by the edge ($1 \le u_i, v_i \le n$).

It is guaranteed that the given graph is a cactus and it contains no multiple edges.

Output

The only line of output must contain one integer $p \cdot q^{-1} \mod 10^9 + 7$. If the process is infinite, output "`NaN`" instead (quotes for clarity).

Examples

standard input	standard output

`6 6 1`
`GRGRGB`
`1 2`
`1 3`
`2 4`
`3 4`
`4 5`
`4 6`
	`250000002`

`14 15 2`
`RGRGRBGGBRBRGG`
`1 2`
`2 3`
`2 4`
`5 2`
`7 4`
`4 6`
`10 6`
`11 7`
`10 13`
`11 13`
`8 5`
`8 12`
`8 9`
`14 9`
`12 14`
	`833333340`

Note

Both samples correspond to the image in the statement. The first sample corresponds to the left cactus and the second sample — corresponds to the right one.

Problem B. Binary Neural Network

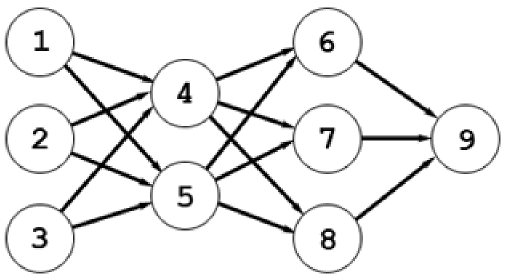
Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 mebibytes

An *artificial neural network*, often called just a *neural network*, is a mathematical model inspired by biological neural networks. A neural network consists of an interconnected group of *artificial neurons*, and it processes information using a connectionist approach to computation.

The neural network is called *layered* if its neurons are organized into groups called *layers*. There is no connection between two neurons belonging to the same layer.

Let us number the neurons with sequential positive integers from 1 to q, where q is the number of neurons in the network. The connection between neuron i and neuron j can be described with the *connection weight* $w_{i,j}$.

A neural network can be represented as a directed acyclic graph: neurons can be represented by the vertices, and connections between neurons — by the edges. The connection weight for each connection can be represented as the weight of the corresponding edge.



In the image above there is a neural network consisting of four layers. The first layer contains neurons 1, 2 and 3, the second layer contains neurons 4 and 5, the third layer consists of neurons 6, 7 and 8, and the fourth one contains the only neuron 9.

Each neuron i has its value v_i, which is calculated by the following formula:

$$v_i = \frac{1}{1 + e^{-\sum_j v_j \cdot w_{j,i}}}$$

The layered neural network is called *binary neural network* if it has the following properties:

- For each neuron i of the first layer the value v_i is either 0 or 1;
- If neuron a belongs to the layer i, neuron b belongs to the layer j and $i > j$, then there is no connection from neuron a to neuron b. Note that the connection from neuron b to neuron a is still possible;
- The last layer has the only neuron and its value is either 0 or 1;
- Each layer contains at least one neuron.

In this problem you are to create a binary neural network that implements the binary function $f(x_1, x_2, \ldots x_n)$ with n arguments.

The first layer of this binary neural network should contain exactly n neurons numbered with sequential positive integers from 1 to n. The value of neuron i will be automatically set to x_i. All the other neurons should be numbered with sequential positive integers from $n + 1$ to q, where q is the number of neurons in the network. All the values v_i ($n + 1 \le i \le q$) will be calculated by the formula that is given above.

The last layer of this binary neural network should contain the only neuron. The value of this neuron should differ from the value $f(x_1, x_2, \dots x_n)$ by no more than 10^{-7}.

The binary network should not contain more than 25 layers. The number q of neurons should not be greater than 10^4. The total number e of connections should not be greater than $3 \cdot 10^4$.

Input

The first line of input contains the only integer n ($2 \le n \le 10$). The second line of input contains 2^n characters. Each of these characters is either '0' or '1'. The first character describes the value of $f(0, \dots, 0, 0)$, the second one describes the value of $f(0, \dots, 0, 1)$ and so on, the last one describes the value of $f(1, \dots, 1, 1)$.

Output

On the first line output two integers l and q — the number of layers and the number of neurons in the binary neural network respectively ($2 \le l \le 25$, $1 \le q \le 10^4$).

On the second line output q integers p_i. Here p_i is the number of layer that the neuron i belongs to ($1 \le p_i \le l$).

On the third line output the only integer e — the number of connections in the binary neural network ($n \le e \le 3 \cdot 10^4$).

Each of the following e lines should contain two integers a_i, b_i and one real number $w_{a,b}$ — description of the connection from a_i to b_i with the weight $w_{a,b}$ ($|w_{a,b}| \le 1000$).

Example

standard input	standard output

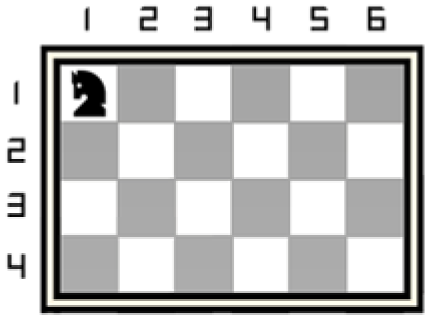
3
00010111
	3 7
	1 1 1 2 2 2 3
	12
	1 4 8.906
	1 5 0.749
	1 6 5.423
	2 4 -0.262
	2 5 -8.905
	2 6 5.582
	3 4 -0.663
	3 5 1.087
	3 6 -12.123
	4 7 66.372
	5 7 -55.329
	6 7 -47.883

Problem C. Chess Puzzle

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

There is a chessboard having four rows and n columns. All rows are numbered with the sequential integers from 1 to 4, and, similarly, all the columns are numbered with sequential integers from 1 to n. Each square of the chessboard can be described with its coordinates $[r, c]$, where r is its row number and c — its column number.

Also, there is a chess knight standing at the square $[1, 1]$. The example of $4 \times n$ chessboard with a knight on it is shown below:



Knight's target is to make a traversal, that starts and ends at the same square $[1, 1]$. Knight should visit each square (except for the square $[1, 1]$) only once during its traversal.

Chess knight can move from some square to a square that is two squares horizontally and one square vertically, or two squares vertically and one square horizontally. The complete move therefore looks like the letter L.

You are to write a program that will find the maximum number of different visited squares of the knight's traversal and this traversal itself.

Input

The only line of input contains the only integer n ($2 \le n \le 10^4$) — the number of columns on the chessboard.

Output

The first line of output should contain the maximum number of squares m that a knight can visit, traversing every square of the chessboard only once (except for the square $[1, 1]$).

The second line of output should contain the coordinates of the squares to visit in order of the optimum traversal. All the coordinates should be printed as $r_1\ c_1\ r_2\ c_2\ \ldots\ r_m\ c_m\ r_1\ c_1$. Coordinates should be printed with no line breaks and should be separated by the only space.

Example

standard input	standard output

`6`	`22`
	`1 1 2 3 1 5 3 6 2 4 1 2 3 1 4 3 3 5 1`
	`4 2 2 4 1 3 3 4 5 2 6 3 4 4 2 2 1 1 3`
	`2 5 4 4 3 2 1 1`

Problem D. Dominoes

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Pasha has a board of size $n \times m$. Some cells are colored black, some — colored green, all remaining cells are colored white. Also, Pasha has a typical domino set consisting of 28 dominoes. He wants to place some of this dominoes on the board in the following way:

- Each domino occupies exactly two neighboring cells;
- Each non-black cell is occupied with exactly one domino;
- There is no black cell occupied with domino;
- The total number of dots on the green cells g is maximized.

You are to write a program to find the maximum possible value of g for the given board.

Input

There will be multiple test cases in the input. Each test case starts with two positive integers n and m ($1 \le n \cdot m \le 56$). The following n lines contain m characters each. The i-th line describes the i-th row of the board: 'W' denotes the white cell, 'B' — the black cell and 'G' — the green cell. Each board contains at least one green cell. The last test case is followed by a line that contains two zeroes. It must not be processed. There will be no more than 500 test cases in the input.

Output

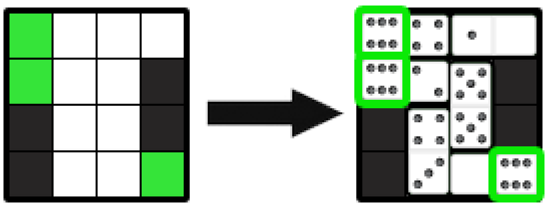
For each test case output its number and then the maximum value of g. If it is impossible to place dominoes in the described way — output "No solution" instead. Follow the format of the sample output.

Example

standard input	standard output
4 4 GWWW GWWB BWWB BWWG 1 3 WGW 0 0	Case 1: 18 Case 2: No solution

Note

The following image shows how to place dominoes in the first sample:



Problem E. Experience is Worth It

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	256 mebibytes

Pasha is playing the video game "DiaBro III". He is fighting against $n \cdot m$ monsters in this game. The monsters are arranged into n rows of m columns each. Rows are numbered with sequential integers from 1 to n and columns — with sequential integers from 1 to m.

Each monster has one of k types. The i-th monster type is described by two integers q_i and g_i: Pasha can kill a monster of i-th type only if amount of his experience is at least q_i, and after killing each monster of this type — Pasha will gain g_i units of experience.

Pasha wants to choose a rectangle by fixing four integers r_1, r_2, c_1 and c_2 such that $1 \le r_1 \le r_2 \le n$ and $1 \le c_1 \le c_2 \le m$. Then Pasha starts to kill monsters in the chosen rectangle — at cells (r, c) such that $r_1 \le r \le r_2$ and $c_1 \le c \le c_2$. Initially, he has no experience at all. The rectangle is *good* if it is possible to kill all monsters inside the rectangle in some order without killing any monster which is not in the rectangle.

You are to write a program that will find the number of different *good* rectangles.

Input

The first line of input contains two integers n and m ($1 \le n, m \le 200$) — the number of rows and columns of monsters respectively.

Each of the following n lines contains exactly m lowercase Latin letters. The j-th character of i-th line denotes the type of monster with row number i and column number j. Monsters of the same type are denoted with the same lowercase Latin letter.

The next line contain the only integer k ($1 \le k \le 26$) — the number of monster types.

Each of the following k lines contains the description of some monster type. The description of i-th monster type consists of character l_i — the lowercase Latin letter corresponding to this monster type — and two integers q_i and g_i ($0 \le q_i \le 10^9, 1 \le g_i \le 10^9$) — the amount of experience required to kill a monster of this type and the amount of experience obtained after killing each monster of this type respectively. The values of l_i, q_i and g_i are separated by single spaces.

It is guaranteed that there is no monster of type that is not described in the input.

Output

The only line of output should contain one integer — the number of ways to choose values r_1, r_2, c_1 and c_2 to satisfy all the conditions given above.

Examples

standard input	standard output
`2 3`	`11`
`aba`	
`baa`	
`2`	
`a 0 2`	
`b 4 100`	
`4 6`	`128`
`aaaaaa`	
`abbaaa`	
`aaacba`	
`aaabba`	
`3`	
`a 0 1`	
`b 3 2`	
`c 12 5`	

Note

There are 11 possible values of r_1, r_2, c_1 and c_2 in the first sample:

1. $r_1 = 1, \quad r_2 = 1, \quad c_1 = 1, \quad c_2 = 1;$
2. $r_1 = 1, \quad r_2 = 1, \quad c_1 = 1, \quad c_2 = 3;$
3. $r_1 = 1, \quad r_2 = 1, \quad c_1 = 3, \quad c_2 = 3;$
4. $r_1 = 1, \quad r_2 = 2, \quad c_1 = 1, \quad c_2 = 2;$
5. $r_1 = 1, \quad r_2 = 2, \quad c_1 = 1, \quad c_2 = 3;$
6. $r_1 = 1, \quad r_2 = 2, \quad c_1 = 2, \quad c_2 = 3;$
7. $r_1 = 1, \quad r_2 = 2, \quad c_1 = 3, \quad c_2 = 3;$
8. $r_1 = 2, \quad r_2 = 2, \quad c_1 = 1, \quad c_2 = 3;$
9. $r_1 = 2, \quad r_2 = 2, \quad c_1 = 2, \quad c_2 = 2;$
10. $r_1 = 2, \quad r_2 = 2, \quad c_1 = 2, \quad c_2 = 3;$
11. $r_1 = 2, \quad r_2 = 2, \quad c_1 = 3, \quad c_2 = 3.$

Problem F. Fix the Matrix

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

This is an interactive problem. In this problem, you have to respond to the actions of the interactor program.

Firstly, you need to output the matrix H of size 6×6 filled with characters '`A`' and '`B`'.

In each of the requests the interactor will perform one of the following actions on matrix H to obtain the new matrix T:

1. Change at most one character in each row of matrix H and reorder its rows in an arbitrary manner.
2. Change at most one character in each column of matrix H and reorder its columns in an arbitrary manner.

Operation of character changing means replacing some character '`A`' with character '`B`' or vice versa.

In each step you are to find out what type of action did the interactor perform. Also, it is necessary to determine for each row or column (depends on the action type) of matrix T its position in the original matrix H.

Note that the **exact order** used by interactor is **required**. For example, if the whole matrix H consists of characters '`A`' and interactor changed no characters but reordered some rows, the output with original or any other order, except for the one that is used by the interactor, would be considered incorrect. That is why it does not make sense to use, for example, matrix H with duplicate rows or columns.

You are to write a program that fills the matrix H and responds correctly to all requests from the interactor program.

It is possible that the same matrix can be obtained by performing action of either the first or the second type. In this case you are allowed to output both these variants, one after another. Your response will be considered correct if at least one variant is identical to that performed by the interactor.

Input

After your program has output matrix H, the interactor will start to provide your program with requests. For each request it will perform some type of action on the original matrix H to obtain the result matrix T. Type of action (first or second) and order for rows or columns can be chosen by interactor depending on both matrix H and test case. Each request starts with a line "`Request`" (quotes for clarity). This line is followed by six lines containing six characters each — the matrix T.

After recieving line "`Accepted`", your program should be terminated immediately.

It is guaranteed that there will be no more than 100 requests and in each of them the interactor will provide you with a correct matrix T.

Output

Firstly your program should output six lines containing six characters each — the initial matrix H. Each character should be either '`A`' or '`B`'.

For each of the following requests from the interactor your program should output either two or three lines.

- If there are two possible actions with different action types that lead to the same matrix your program should output three lines. On the first line there should be the only integer 0 (zero) denoting that there are two possible action types. The second line should describe the possible action of the first type and contain six space-separated integers — i-th integer is the index of i-th row of matrix T in the original matrix H. Similarly, the third line should describe the possible action of the second type and contain six space-separated integers — i-th integer is the index of i-th column of matrix T in the original matrix H.

- If there is the only possible action type your program should output two lines. On the first line there should be the only integer t — the type of action performed by the interactor. The second line should contain six space-separated integers — i-th integer is the index of i-th row or column (depends on the action type) of matrix T in the original matrix H.

All rows are numbered with sequential integers from 1 to 6 from top to bottom, all columns are numbered in the same way from left to right.

Example

standard input	standard output
<waiting for output>	AAAAAA
<waiting for output>	BBBBBB
<waiting for output>	AAABBB
<waiting for output>	ABABAB
<waiting for output>	BABABA
<waiting for output>	BBAAAA
Request	*<reading input>*
AAABAA	*<reading input>*
BBAAAB	*<reading input>*
BBBBBB	*<reading input>*
BABABB	*<reading input>*
ABABAB	*<reading input>*
AAABBA	*<reading input>*
<waiting for output>	1
<waiting for output>	1 6 2 5 4 3
Request	*<reading input>*
AABAAA	*<reading input>*
BBBBBB	*<reading input>*
AAABBB	*<reading input>*
AABBAB	*<reading input>*
BBAABA	*<reading input>*
BABAAA	*<reading input>*
<waiting for output>	2
<waiting for output>	1 3 2 4 5 6
Accepted	*<reading input and terminating>*

Problem G. Guess the Data Structure

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	256 mebibytes

You are given an array A consisting of n integers. All its elements are numbered with consecutive integers from 1 to n. Your task is to process m queries. Each of these queries is one of the following kinds:

1. Append integer x_j to the end of the array A.
2. Output the value of $\sum_{i=l_j}^{r_j} A_i$ $(l_j \le r_j)$.
3. Perform $A_i = A_i \oplus x_j$ for each element of the array A. Expression $A_i \oplus x_j$ means applying the operation of bitwise exclusive "OR" to numbers A_i and x_j.
4. Sort the array A.

Your task is to implement a proper data structure to process m given queries for a given array A.

Input

First line of input contains the only integer n — the initial size of the array A $(1 \le n \le 2 \cdot 10^5)$.

Second line of input contains n non-negative integers A_i — elements of the array A $(0 \le A_i \le 10^9)$.

Third line of input contains the only integer m — the number of queries $(1 \le m \le 2 \cdot 10^5)$.

Each of the following lines contains a separate query in the following format:

- For the first kind of query: 1 x_j
- For the second kind of query: 2 l_j r_j
- For the third kind of query: 3 x_j
- For the fourth kind of query: 4

For any query of the first or the third kind $0 \le x_j \le 10^9$. For any query of the second kind l_j and r_j do not violate bounds of the array A.

You may assume that at least one query in each test case is of the second kind.

Output

For each query of the second kind output its resulting sum on a separate line.

Example

standard input	standard output

5
3 4 7 3 6
7
3 5
2 2 4
4
1 15
2 3 6
3 1
2 1 6
	9
	30
	33

Problem H. Hovercraft

Input file:	*standard input*
Output file:	*standard output*
Time limit:	8 seconds
Memory limit:	256 mebibytes

You are controlling a hovercraft. Hovercraft is located on a secret base. This base is divided into $n \times m$ sectors. Rows are numbered with consecutive integers from 1 to n (from top to bottom), columns — from 1 to m (from left to right). Each sector is either empty or occupied with a wall. Also, there are k commutators located at some empty sectors. There is no sector with more than one commutator in it. Each commutator can be either enabled or disabled.

You need to make a list of 12 commands to the hovercraft, and it will execute these commands one by one in the same order. Each of these commands should be one of a following type:

- `U` — move up: move from sector $[r, c]$ to sector $[r-1, c]$;
- `D` — move down: move from sector $[r, c]$ to sector $[r+1, c]$;
- `L` — move left: move from sector $[r, c]$ to sector $[r, c-1]$;
- `R` — move right: move from sector $[r, c]$ to sector $[r, c+1]$;
- `S` — switch commutator state: enable commutator if it was disabled, and disable otherwise;
- `F` — call function: execute a predefined list of 8 commands one by one;
- `N` — no operation: just do nothing.

Hovercraft ignores movement operation if it is impossible to perform it without hitting a wall or violating the secret base borders.

Hovercraft ignores switching operation if there is no commutator in the current cell.

Each function call has the same list of commands, and it is also composed by you. This list should contain exactly 8 commands of the above types. It means that the function can be called recursively.

You goal is to make all k commutators enabled at some moment of time. That means you task is to compose a list of 12 commands and a function of 8 commands to make hovercraft enable all k commutators at the same time. It does not matter what will happen after this moment of time: hovercraft can move indefinitely or even disable some commutators.

Input

The first line of input contains three integers n, m and k — the number of rows and columns of the secret base and the number of commutators respectively ($2 \le n, m \le 20$, $1 \le k \le 5$).

The second line of input contains two integers x_s and y_s — row and column number of the initial hovercraft position ($1 \le x_s \le n$, $1 \le y_s \le m$).

Each of the following n lines denotes a row of the secret base and contains m characters. Each character is either '`.`' (empty sector), '`X`' (sector with wall), '`E`' (enabled commutator) or '`D`' (disabled commutator).

It is guaranteed that the hovercraft starts in a sector without a wall, and also that at least one valid answer exists for the given input.

Output

The first line of output should contain exactly 8 characters denoting commands of function.

The second line of output should contain exactly 12 characters denoting the main list of commands.

Each of these characters should be either 'U', 'D', 'L', 'R', 'S', 'F' or 'N'.

If there are several possible answers, output any one of them.

Examples

standard input	standard output

`5 6 3`
`1 3`
`.E....`
`XXXXX.`
`...D..`
`.XXXXX`
`.....D`
	`UUUDDRSF`
	`RRRDDLLSLLLF`

`3 4 2`
`3 3`
`E...`
`.XX.`
`.X.E`
	`NNNNNNNN`
	`RSUULLLSNNNN`

`4 6 3`
`4 6`
`DX....`
`.X.XX.`
`.D..X.`
`....XD`
	`NSLUUUSN`
	`NFLLNLDDNLFS`

Note

There is no need to use function in the second sample. In this case the function can be composed of arbitrary commands.

After performing the second function call of the third sample, all the commutators are enabled. That is why the output is correct despite the fact that the last command disables one of the commutators.

Problem I. Izhevsk Training Camp

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	256 mebibytes

Izhevsk Training Camp is about to begin! This season n teams numbered with consecutive integers from 1 to n will take part in this event. Nine sophisticated contests will be offered to participating teams in eleven days period. Contest are numbered with consecutive integers from 1 to 9. Three of these contests will form the Udmurtia Head Super Cup (UHSC). The question is: what three contests to choose for UHSC?

Oleg is helping to hold the Izhevsk Training Camp. For any contest he knows in advance what place each of the n teams will take in this contest. He wants to use his knowledge to select three contests in order to minimize the total *boredom* of UHSC.

The *boredom* of UHSC can be computed as number of pairs of teams $\{i, j\}$ such that team i won team j in each of three UHSC contests.

You are to write a program that will help Oleg to find three contests a, b and c for UHSC such that the total *boredom* of UHSC is minimum possible.

Input

The first line of input contains an integer n — the number of participating teams ($2 \le n \le 2^{16}$).

The i-th of the following nine lines contains the i-th contest description: n unique positive integers from 1 to n — team numbers ordered from the first place to the last.

Output

The only line of output should contain three positive integers a, b and c — numbers of contests to choose for UHSC ($1 \le a, b, c \le 9$, $a \ne b, a \ne c, b \ne c$).

If there are multiple correct answers — output any of them.

Example

standard input	standard output

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

Note

For the sample test case the minimum possible value of *boredom* is 5.