Problem 155. Cartesian Tree

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 megabytes

Let us consider a special type of binary search trees, called *cartesian trees*. Recall that a binary search tree is a rooted ordered binary tree, such that for its every node x the following condition is satisfied: each node in its left subtree has the key less than the key of x, and each node in its right subtree has the key greater than the key of x.

That is, if we denote the left subtree of the node x by $L(x)$, its right subtree by $R(x)$ and its key by k_x, for each node x we will have

- if $y \in L(x)$ then $k_y < k_x$
- if $z \in R(x)$ then $k_z > k_x$

The binary search tree is called *cartesian* if its every node x in addition to the main key k_x also has an auxiliary key that we will denote by a_x, and for these keys the *heap condition* is satisfied, that is

- if y is the parent of x then $a_y < a_x$

Thus a cartesian tree is a binary rooted ordered tree, such that each of its nodes has a pair of two keys (k, a) and three conditions described are satisfied.

Given a set of pairs, construct a cartesian tree out of them, or detect that it is not possible.

Input

The first line of the input file contains an integer number N — the number of pairs you should build cartesian tree out of ($1 \le N \le 50\,000$). The following N lines contain two integer numbers each — given pairs (k_i, a_i). For each pair $|k_i|, |a_i| \le 30\,000$. All main keys and all auxiliary keys are different, i.e. $k_i \ne k_j$ and $a_i \ne a_j$ for each $i \ne j$.

Output

On the first line of the output file print `YES` if it is possible to build a cartesian tree out of given pairs or `NO` if it is not. If the answer is positive, output the tree itself in the following N lines. Let the nodes be numbered from 1 to N corresponding to pairs they contain as these pairs are given in the input file. For each node output three numbers — its parent, its left child and its right child. If the node has no parent or no corresponding child, output `0` instead.

If there are several possible trees, output any one.

Example

standard input	standard output

7
5 4
2 2
3 9
0 5
1 3
6 6
4 11
	YES
	2 3 6
	0 5 1
	1 0 7
	5 0 0
	2 4 0
	1 0 0
	3 0 0

Problem 156. Strange Graph

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 megabytes

Let us consider an undirected graph $G = \langle V, E\rangle$. We will say that two vertices u and v are *neighbours* if $(u, v) \in E$. In this case we also say that u is a neighbour of v and v is a neighbour of u. Let us denote by $N(v)$ the set of neighbours of v. Recall that the number of neighbours of v is called the *degree* of this vertex and is denoted by $\deg v$.

We will call graph G *strange* if it is connected and for its every vertex v the following conditions are satisfied:

1. $\deg v \geq 2$ (i.e. there are at least two neighbours of v)

2. if $\deg v = 2$ then the two neighbours of v are not connected by an edge

3. if $\deg v > 2$ then there is $u \in N(v)$, such that the following is true:

   (a) $\deg u = 2$

   (b) any two different vertices $w_1, w_2 \in N(v) \setminus \{u\}$ are neighbours, i.e. $(w_1, w_2) \in E$

You are given some strange graph G. Find hamiltonian cycle in it, i.e. find such cycle that it goes through every vertex of G exactly once.

Input

The first line of the input file contains two integer numbers N and M — the number of vertices and edges in G respectively ($3 \leq N \leq 10\,000$, $M \leq 100\,000$). $2M$ integer numbers follow — each pair represents vertices connected by the corresponding edge (vertices are numbered from 1 to N). It is guaranteed that each edge occurs exactly once in the input file and that there are no loops (i.e. ends of each edge are distinct).

Output

If there is no hamiltonian cycle in G, print `-1` in the first line of the output file. In the other case output N numbers — the sequence of vertices of G as they appear in the hamiltonian cycle found (note that the last vertex must be connected to the first one). If there are several solutions, output any one.

Example

standard input	standard output

`4 4`
`1 2  2 3  3 4  4 1`
	`1 2 3 4`

`9 12`
`1 2  2 3  3 1  1 4  2 5  3 6`
`4 7  5 8  6 9  7 8  8 9  9 7`
	`-1`

Problem 157. Patience

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 megabytes

Sasha enjoys playing patiences. Last days he played one very interesting patience on a rectangular grid 14×4. The rules of this patience are very simple. He takes a standard 52-card deck (without jokers), extracts all aces and lays them in the first column of the grid. Cell $(1, 1)$ contains ace of diamonds, $(1, 2)$ — ace of hearts, $(1, 3)$ — ace of clubs and $(1, 4)$ contains ace of spades. Then Sasha shuffles the rest of the deck and lays remaining cards to the grid row by row leaving column 2 empty, so all the columns from 3 to 14 are covered by cards.

After having laid all cards in such a manner, he is allowed to make moves. The move is determined by selection of a free cell. This free cell is to be covered by the card defined by the contents of the left adjacent cell. The covering card must have the same suit and the next higher value (order of values is: Ace 2 3 4 5 6 7 8 9 10 Jack Queen King). For example, if the cell $(6, 3)$ contains the Queen of Spades, and the cell $(7, 3)$ is empty, selecting cell $(7, 3)$ will move the King of Spades to $(7, 3)$ (and leave empty the cell where the King of Spades was before the move). If the left neighbour of the free cell contains a King or is empty, the move selecting this free cell is disabled.

The goal is to make each row ordered (i.e. columns from 1 to 13 must contain cards from Ace to King, and the column 14 must be empty). If all free cells follow Kings or empty cells, and the cards are not ordered, Sasha is declared a loser (in the classic version of this patience, the rest of the deck may be shuffled two times again leaving ordered cards on their positions, however Sasha is very experienced and often wins without additional shuffling).

Now Sasha wants to know some things about this patience. Imagine the position where three suits are already ordered, and the remaining suit is ordered partially. It is known that there are less than n highest cards that are not ordered. So only n rightmost columns may contain unordered cards. For example, if $n = 3$, columns from 1 to 11 are ordered, and columns from 12 to 14 contain Queen, King and empty cell in an arbitrary order. Sasha wants to determine how many of such positions for a given n have a winning strategy. For example, if $n = 3$, there is exactly one position that does not allow him to win: ... [King] [empty] [Queen]. All other positions have a winning strategy:

1. ... [Queen] [King] [empty] is already a goal position
2. ... [Queen] [empty] [King] allows to move King adjacent to the Queen.
3. ... [empty] [Queen] [King] becomes previous position after moving the Queen after the Jack.
4. ... [empty] [King] [Queen] becomes goal position by moving the Queen after the Jack.
5. ... [King] [Queen] [empty] becomes position 3 after moving the King to the free cell.

So for $n = 3$ the answer is 5.

Write a program that will get the answer for an arbitrary n from 1 to 13.

Input

Input file consists of only one integer number n.

Output

Output the number of positions with less than n highest cards not ordered in the remaining suit that have a winning strategy.

Example

standard input	standard output
3	5
4	14

Problem 158. Commuter Train

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 megabytes

You might have noticed that bus drivers sometimes pass by passengers waiting for the bus and stop at a place where the distance from the people to the bus doors is maximal. We don't know exact reasons of such behaviour, probably, this is done not because of some special wickedness of the driver but in order to allow passengers in the vehicle to get off quicker. However, we know that in one country the government decided to implement automatic driving system on their commuter railroads. Among other features, this system is intended to automatically stop commuter trains at stations. A computer in the train is connected to a special radar, which determines passenger positions at the station platform. The computer decides where to stop the train in order to maximize the sum of the distances from each passenger to the closest door. All the hardware is ready, but the software project is late (transport problems). Your task is to implement this function for this software project.

More precisely, the station platform has length $0 < L \leq 5000$. There are $0 < M \leq 300$ passengers at the platform. Each passenger p has position P_p ($0 \leq P_1 \leq ... \leq P_M \leq L$) — the distance from the platform beginning to the passenger. There are $0 < N \leq 300$ doors in the train. Each door d has position D_d ($0 = D_1 < D_2 < ... < D_N \leq L$) — the distance from it to the door 1. The door width and the passenger sizes are not taken into account. For simplicity assume that the distance between a passenger i and a train door j is $dist(i, j, S) = |D_j + S - P_i|$, where S is train position — the distance between the first door and the beginning of the platform. Remember that the train must stop so that no door is outside of the platform.

Input

The file contains integer numbers separated by spaces and/or line feeds. At the beginning of the file there is the station description (L, M, and P_1 ... P_M), followed by the train description (N and D_2 ... D_N).

Output

You should output two numbers — the train position S, at which the maximal possible sum of the minimal distances between the passengers and the doors can be achieved and the sum itself. If there are many such train positions — output any one.

Example

standard input	standard output

6
2
0 4
2
4
	2 4

4
5
0 1 2 3 4
4
1 2 3
	0.5 2.5

Problem 159. Self-Replicating Numbers

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 megabytes

Vasya's younger brother Misha is fond of playing with numbers. Two days ago he discovered that $9376^2 = 87909376$ — the last four digits constitute 9376 again. He called such numbers *self-replicating*. More precisely, an n-digit number is called *self-replicating* if it is equal to the number formed by the last n digits of its square. Now Misha often asks Vasya to help him to find new such numbers. To make the things worse, Vasya's brother already knows what the scales of notation are, so he asks Vasya to find, for example, hexadecimal or binary self-replicating numbers.

Vasya wants to help his brother, but unfortunately he is very busy now: he is seriously preparing and training for the next ACM Regional Contest. So he asked *you* to write a program that for a given base b and length n will find all n-digit self-replicating numbers in the scale of notation with base b.

Input

The only line of the input contains two integer numbers b and n separated by a single space, the base b of the scale of notation ($2 \le b \le 36$) and the required length n ($1 \le n \le 2000$).

Output

The first line of the output contains K — the total number of self-replicating numbers of length n in base b. Next K lines contain one n-digit number in base b each. Uppercase Latin letters from A to Z must be used to represent digits from 10 to 35. The self-replicating numbers can be listed in arbitrary order.

Example

`standard input`	`standard output`

`10 4`	`1`
	`9376`

`12 6`	`2`
	`1B3854`
	`A08369`

Problem 160. Magic Multiplying Machine

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 megabytes

Well known Las-Vegas casino “Big Jo” has recently introduced the new playing machine, called *Magic Multiplying Machine* (MMM). MMM has N levers and one big red button. Each lever is marked with some integer number ranging from 1 to M, thus i-th lever is marked with number a_i.

A player who wishes to play on MMM inserts a coin into a special slot. After that she chooses some levers and pulls them. After she has made her choice, she presses the big red button. MMM blinks all its lights, rings its bells, plays different tunes and after that declares whether the player has won the game.

The algorithm for determining whether the player has won is the following. If the player has pulled some subset $S \subset \{1, 2, \ldots, N\}$ of levers, her score is the product of numbers that these levers are marked with, taken modulo M (if the player has pulled no levers, her score is 1):

$$\mathit{score} = \prod_{i \in S} a_i \bmod M$$

The player has won the game if her score is maximal possible for the given machine.

Given the description of some MMM determine which levers should be pulled to win the game.

Input

The first line of the input file contains two integer numbers $1 \le N \le 10\,000$ and $2 \le M \le 1\,000$. The second line contains N integer numbers $a_1, a_2, \ldots, a_N$ ranging from 1 to M.

Output

In the first line of the output file print one integer number — the score that should be obtained to win the game. In the second line print in ascending order the numbers of levers that should be pulled to obtain this score (levers are numbered starting from one).

If there are several solutions, output any one.

Example

standard input	standard output

`4 6`
`1 2 3 4`
	`4`
	`1 4`

`4 4`
`4 4 4 4`
	`1`

Problem 161. Intuitionistic Logic

Input file:	`standard input`
Output file:	`standard output`
Time limit:	15 seconds
Memory limit:	64 megabytes

Recently Vasya became acquainted with an interesting movement in mathematics and logic called "intuitionism". The main idea of this movement consists in the rejection of the law of excluded middle (the logical law stating that any assertion is either true or false). Vasya liked this idea; he says: "Classical mathematics says that Fermat Last Theorem is either true or false; but this statement is completely useless for me until I see the proof or a contrary instance". So Vasya became a born-again intuitionist. He tries to use the intuitionistic logic in all his activities and especially in his scientific work. But this logic is much more difficult than the classical one. Vasya often tries to use logical formulae that are valid in classical logic but aren't so in the intuitionistic one.

Now he wants to write a program that will help him to check the validity of his formulae automatically. He has found a book describing how to do that but unfortunately he isn't good at programming, so you'll have to help him.

The construction starts from an arbitrary acyclic oriented graph $\mathfrak{X} = \langle X, G\rangle$. Then a partial order is constructed on X, the set of vertices of $\mathfrak{X}$: for any $x, y \in X$ we define $x \leq y$ iff there exists a path (possibly of zero length) in $\mathfrak{X}$ from x to y. Next, consider the set $\mathfrak{P}$ of all subsets of X and the set $\mathcal{H} \subset \mathfrak{P}$ consisting of all $\alpha \subset X$ such that any two different x and y from α are incomparable (i.e. neither $x \leq y$ nor $y \leq x$). Note that $\mathcal{H}$ always contains the empty set and all one-element subsets of X. Now it is possible to define a map $\mathrm{Max} : \mathfrak{P} \to \mathcal{H} \subset \mathfrak{P}$. For any $M \subset X$ we put $\mathrm{Max}(M) = \{x \in M : \nexists y \in M : x \neq y, x \leq y\}$ — the set of all maximal elements of M.

Next we define several operations on $\mathcal{H}$. For any $\alpha, \beta \in \mathcal{H}$ put $\alpha \Rightarrow \beta = \{x \in \beta : \nexists y \in \alpha : x \leq y\}$, $\alpha \wedge \beta = \mathrm{Max}(\alpha \cup \beta)$, $\alpha \vee \beta = \mathrm{Max}(\{x \in X : \exists y \in \alpha, z \in \beta : x \leq y, x \leq z\})$, $\mathbf{0} = \mathrm{Max}(X)$, $\mathbf{1} = \varnothing$, $\neg\alpha = (\alpha \Rightarrow \mathbf{0})$, $\alpha \equiv \beta = ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$.

Now consider logical formulae consisting of the following symbols:

- Constants **1** and **0**;
- Variables — capital letters from A to Z;
- Parentheses — if E is a formula, then (E) is another;
- Negation — $\neg E$ is a formula for any formula E;
- Conjunction — $E_1 \wedge E_2 \wedge \cdots \wedge E_n$. Note that the conjunction is evaluated from left to right: $E_1 \wedge E_2 \wedge E_3 = (E_1 \wedge E_2) \wedge E_3$.
- Disjunction — $E_1 \vee E_2 \vee \cdots \vee E_n$. The same remark applies.
- Implication — $E_1 \Rightarrow E_2$. Unlike the previous two operations it is evaluated from right to left: $E_1 \Rightarrow E_2 \Rightarrow E_3$ means $E_1 \Rightarrow (E_2 \Rightarrow E_3)$.
- Equivalence — $E_1 \equiv E_2 \equiv \cdots \equiv E_n$. This expression is equal to $(E_1 \equiv E_2) \wedge (E_2 \equiv E_3) \wedge \cdots \wedge (E_{n-1} \equiv E_n)$.

The operations are listed from the highest priority to the lowest.

A formula E is called *valid* (in the model defined by $\mathfrak{X}$) if after substitution of arbitrary elements of $\mathcal{H}$ for the variables involved in E it evaluates to **1**; otherwise it is called *invalid.*

Your task is to determine for a given graph $\mathfrak{X}$ which formulae from a given set are valid and which invalid.

Input

The first line contains two integers N and M separated by a single space — the number of vertices ($1 \le N \le 100$) and edges ($0 \le M \le 5000$) of $\mathfrak{X}$. The next M lines contain two integers s_i and t_i each — the beginning and the end of i-th edge respectively. The next line contains K ($1 \le K \le 20$) — the number of formulae to be processed. The following K lines contain one formula each. A formula is represented by a string consisting of tokens `0`, `1`, `A`, ..., `Z`, `(`, `)`, `~`, `&`, `|`, `=>`, `=`. The last five tokens stand for $\neg$, $\wedge$, $\vee$, $\Rightarrow$ and $\equiv$ respectively. Tokens can be separated by an arbitrary number of spaces. No line will be longer than 254 characters. All formulae in the file will be syntactically correct. Also you may assume that the number $H = |\mathcal{H}|$ of elements of $\mathcal{H}$ doesn't exceed 100 and that $\sum_{1 \le j \le K} H^{v_j} \le 10^6$ where v_j is the number of different variables used in j-th formula.

Output

The output file must contain K lines — one line for each formula. Write to the j-th line of output either `valid` or `invalid`.

Example

standard input	standard output
1 0 6 1=0 X|~X A=>B=>C = (A&B)=>C ~~X => X X => ~~X (X => Y) = (Y | ~X)	invalid valid valid valid valid valid
6 6 1 2 2 3 2 4 3 5 4 5 5 6 11 1=0 X|~X A=>B=>C = (A&B)=>C ~~X => X X => ~~X (X => Y) = (Y | ~X) A&(B|C) = A&B|A&C (X=>A)&(Y=>A) => X|Y=>A X = ~~X ~X=~~~X ~X = (X => 0)	invalid invalid valid invalid valid invalid valid valid invalid valid valid

Problem 162. Pyramids

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 megabytes

Recently in Farland, a country in Asia, a famous scientist Mr. Log Archeo has discovered ancient pyramids. But unlike those in Egypt and Central America, they have triangular (not rectangular) foundation. That is, they are tetrahedrons in mathematical sense. In order to find out some important facts about the early society of the country (it is widely believed that the pyramid sizes are in tight connection with Farland ancient calendar), Mr. Archeo needs to know the volume of the pyramids. Unluckily, he has reliable data about their edge lengths only. Please, help him!

Input

The file contains six positive integer numbers not exceeding 1000 separated by spaces, each number is one of the edge lengths of the pyramid $ABCD$. The order of the edges is the following: AB, AC, AD, BC, BD, CD.

Output

A real number — the volume printed accurate to four digits after decimal point.

Example

`standard input`	`standard output`
`1 1 1 1 1 1`	`0.1179`
`1000 1000 1000 3 4 5`	`1999.9937`

Problem 163. Wise King

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 megabytes

Once upon a time in a country far away lived a king and he had a big kingdom. He was a very wise king but he had one weakness — he could count only up to three.

Nevertheless, he did not consider this to be a really great drawback, since he had a lot of wizards who could count up to one hundred (and some of them, people said, even up to one thousand). But one day the grief came to the kingdom as the outnumbering barbarians started to approach from all sides. And the king then had to make the most important decision in his life. He had to choose which of his sons to make generals that he would send to the borders of the country to lead the army.

However, the king knew that though some of his sons were smart, just like he was, some of them were quite stupid and could only lower army spirits with their wrong decisions. More precisely, he knew about each of his sons his *mental potential* — an integer number ranging from minus three to three (remember, that the king could count only up to three). He also knew that the chance of his army defeating barbarians was proportional to the sum of some powers of mental potentials of those of his sons that he would make generals (the power exponent was a positive integer number, the same for all his sons and not exceeding three either). Thus he had to choose such a combination of his sons to lead the army, that this sum would be maximal possible.

However, the king himself could not make all appropriate calculations since, for example, the second power (the square) of a number not exceeding three could be greater than three, and therefore he asked you, his most intelligent wizard, to solve this problem.

Input

The first line of the input file contains the number of the sons of the king (integer number less than or equal to one hundred). The second line contains the positive integer number not exceeding three, the exponent in the formula used to calculate the chance of defeating barbarians. The third line contains the list of mental potentials of king's sons — all integer numbers, not greater than three by their absolute value.

Output

Output the only number — the maximal possible chance of defeating barbarians calculated as the sum described.

Example

In the first example below the king should choose his first and third sons to be the generals. In this case the chance to defeat barbarians, which is the sum of cubes of mental potentials of these sons, is eight plus one, that is nine.

In the second example sending his son to lead the army causes the sum to be negative, thus he should not do it and the sum would be zero.

standard input	standard output

3
3
2 -1 1
	9

1 1 -1	0