Interval Set Cover

Alice has an array of n empty sets: $A[1], A[2], \ldots, A[n]$. Since the empty sets are not interesting, she wants to perform k operations on them. On operation i, Alice selects two integers $l[i]$ and $r[i]$ such that $1 \le l[i] \le r[i] \le n$, and adds the element i to all sets $A[j]$ such that $l[i] \le j \le r[i]$.

For example, if $n = 3$ and $k = 2$, and Alice picks $(l[1], r[1]) = (2, 3)$ and $(l[2], r[2]) = (1, 3)$, then, after the first operation,

$$A[1] = \{\}, \quad A[2] = \{1\}, \quad \text{and} \quad A[3] = \{1\},$$

and after the second operation,

$$A[1] = \{2\}, \quad A[2] = \{1, 2\}, \quad \text{and} \quad A[3] = \{1, 2\}.$$

After all k operations are done, Alice's sets have a really cool structure. However, Farhan is jealous of this, so he concocts an wicked plan:

- First, he chooses a permutation $p = (p_1, p_2, \ldots, p_n)$ of the integers $1, 2, \ldots, n$; that is, p has each of the numbers from 1 to n exactly once in some order of Farhan's choosing.
- Then he creates another array B of sets $B[1], B[2], \ldots, B[n]$ such that $B[i] = A[p_i]$.
- Finally, he masterfully replaces Alice's array A with his new array B when she is not around.

Alice, of course, notices that her array of sets has been messed around. Unfortunately, she no longer remembers the integers $l[i]$ and $r[i]$. It makes Alice sad, so she wants to keep some of sets from the new array and throw the rest of them away. But she wants to do it in a such that every element from 1 to k is in at least of one the sets that she keeps.

More formally, Alice wants to select the minimum number of sets such that their union is $\{1, 2, \ldots, k\}$.

Input

You will be given the array of sets B in the input. Let's assume that the set $B[i]$ has c_i unique elements. Let those elements be $B[i][1], B[i][2], \ldots, B[i][c_i]$ **without duplication**.

Read the input from the standard input in the following format:

- line 1: $\;n \;\; k$
- line $2i$ ($1 \le i \le n$): $\;c_i$
- line $2i + 1$ ($1 \le i \le n$): $\;B[i][1] \;\; B[i][2] \;\; \ldots \;\; B[i][c_i]$

Output

Suppose the minimum number of sets Alice has to keep is m, and the m sets she chooses are $B[t_1], B[t_2], \ldots, B[t_m]$. Write the output to the standard output in the following format:

- line 1: m
- line 2: $t_1 \; t_2 \; \ldots \; t_m$

You can output the indices in any order. If there are multiple solutions, you may print any of them.

Constraints

- $1 \leq n \leq 2000$
- $1 \leq k \leq 2000$
- $1 \leq t_i \leq k$ (for all $1 \leq i \leq n$)
- $1 \leq B[i][j] \leq k$ (for all $1 \leq i \leq n$ and $1 \leq j \leq t_i$)
- For each integer j such that $1 \leq j \leq k$, there is at least one set which contains j.

Subtasks

1. (4 points) $n \leq 15$
2. (5 points) $k \leq 15$
3. (11 points) $p_i = i$ (for all $1 \leq i \leq n$)
4. (17 points) $n, k \leq 80$, and for any two i, j ($1 \leq i, j \leq k$), the ranges $[l[i], r[i]]$ and $[l[j], r[j]]$ must either be disjoint, or one must be contained by another.
5. (20 points) $n, k \leq 80$
6. (15 points) $n, k \leq 300$
7. (28 points) No further constraints.

Example 1

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

The correct output is:

```
1
1
```

This example is described in the problem statement. Notice that $p = (2, 1, 3)$. The optimal solution chooses the set $B[1]$ which includes all elements. Choosing the set $B[3]$ is also a valid solution.

Example 2

```
3 5
2
1 2
2
2 3
3
1 4 5
```

The correct output is:

```
2
2 3
```

Here, it's optimal to keep sets $B[2]$ and $B[3]$, which covers all 5 elements. It's easy to see that there cannot be any solution that includes only one set, hence this is optimal.