A. Defragmentation

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

In this problem you have to implement an algorithm to defragment your hard disk. The hard disk consists of a sequence of clusters, numbered by integers from 1 to n. The disk has m recorded files, the i-th file occupies clusters with numbers $a_{i,1}$, $a_{i,2}$, ..., a_{i,n_i}. These clusters are not necessarily located consecutively on the disk, but the order in which they are given corresponds to their sequence in the file (cluster $a_{i,1}$ contains the first fragment of the i-th file, cluster $a_{i,2}$ has the second fragment, etc.). Also the disc must have one or several clusters which are free from files.

You are permitted to perform operations of copying the contents of cluster number i to cluster number j (i and j must be different). Moreover, if the cluster number j used to keep some information, it is lost forever. Clusters are not cleaned, but after the defragmentation is complete, some of them are simply declared unusable (although they may possibly still contain some fragments of files).

Your task is to use a sequence of copy operations to ensure that each file occupies a contiguous area of memory. Each file should occupy a consecutive cluster section, the files must follow one after another from the beginning of the hard disk. After defragmentation all free (unused) clusters should be at the end of the hard disk. After defragmenting files can be placed in an arbitrary order. Clusters of each file should go consecutively from first to last. See explanatory examples in the notes.

Print the sequence of operations leading to the disk defragmentation. Note that **you do not have to minimize** the number of operations, but it should not exceed $2n$.

Input

The first line contains two integers n and m ($1 \le n, m \le 200$) — the number of clusters and the number of files, correspondingly. Next m lines contain descriptions of the files. The first number in the line is n_i ($n_i \ge 1$), the number of clusters occupied by the i-th file. Then follow n_i numbers $a_{i,1}$, $a_{i,2}$, ..., a_{i,n_i} ($1 \le a_{i,j} \le n$). It is guaranteed that each cluster number occurs not more than once and , that is, there exists at least one unused cluster. Numbers on each line are separated by spaces.

Output

In the first line print a single integer k ($0 \le k \le 2n$) — the number of operations needed to defragment the disk. Next k lines should contain the operations' descriptions as "i j" (copy the contents of the cluster number i to the cluster number j).

Examples

input

```
7 2
2 1 2
3 3 4 5
```

output

```
0
```

input

```
7 2
2 1 3
3 2 4 5
```

output

```
3
2 6
```

```
3 2
6 3
```

Note

Let's say that a disk consists of 8 clusters and contains two files. The first file occupies two clusters and the second file occupies three clusters. Let's look at examples of correct and incorrect positions of files after defragmentation.

Example 2: each file must occupy a contiguous area of memory.

Example 3: the order of files to each other is not important, at first the second file can be written, and then — the first one.

Example 4: violating the order of file fragments to each other is not allowed.

Example 5: unused clusters should be located at the end, and in this example the unused clusters are 3, 7, 8.