Depot Rearrangement

Input file: `depot.in` **100 points**
Output file: `depot.out` **Time limit: 2 sec**
Source Code: `depot.pas/.c/.cpp` **Memory limit: 64 MB**

A company operates ***N*** shops, selling ***M*** different products in each shop. The company has a large depot where the products are packed before delivering to shops. Each shop receives the same number of items of each product. Hence the company packs a certain number of items of a given product into a container, and labels that container with the product identifier. Products are identified by the numbers from ***1*** to ***M***. Thus, at the end of packing, there are ***N*M*** containers in the depot, and exactly ***N*** containers are labeled with a given product label for each product. Because the depot is in a narrow building, the containers are arranged in a single row. In order to speed-up distribution, the manager of the depot wants to rearrange the containers. Since the product delivery to the shops occurs by sending exactly one truck to each shop, and each truck carries one container of each product, a suitable arrangement is the following. The first ***M*** containers in the row must be labeled with different product labels, the second ***M*** containers in the row must be labeled with different product labels, and so on. Unfortunately, there is only one free place at the end of the row to hold a container. Therefore the rearrangement must be performed by successively picking up a container and moving it to the free place. After the rearrangement the free place must be at the end of the row.

The goal is to achieve the required rearrangement by a minimal number of moves.

Task

You are to write a program that computes a rearrangement which needs the minimal number of moves.

Input

The first line of the text file `depot.in` contains two integers, ***N*** and ***M***. ***N*** ($1 \le N \le 400$) is the number of shops and ***M*** ($1 \le M \le 400$) is the number of products. The second line contains ***N*M*** integers, the labels of the containers in their initial order. Each product identifier ***x*** ($1 \le x \le M$) occurs exactly ***N*** times in the line.

Output

The first line of the text file `depot.out` contains one integer ***S***, the minimal number of moves that are necessary to obtain a required order of the container row (Subtask A). The following ***S*** lines describe a rearrangement (Subtask B). Each line contains a pair of integers ***x y***. The pair ***x y*** describes a move: the container at position ***x*** is to move to position ***y***. Positions are identified by the numbers from ***1*** to ***N*M+1***; initially the position ***N*M+1*** is free (holds no container). A move from ***x*** to ***y*** is legal only if position ***y*** is free prior to the move. After a move from ***x*** to ***y*** the position ***x*** will be free. It is enough to output only the first line if you solve only Subtask A.

If there are multiple possibilities, your program should output only one; it does not matter which one.

Example

depot.in	depot.out
5 6	8
4 1 3 1 6 5 2 3 2 3 5 6 2 1 4 5 6 4 1 3 2 4 5 5 1 2 3 4 6 6	9 31
	18 9
	10 18
	4 10
	31 4
	30 31
	24 30
	31 24