Fixing the games

In Semester One of 2021 one of the group études included writing a strategy for the game *For Sale*. We ran a tournament between the different submissions to compare them. Annoyingly, there were seven groups and the maximum number of players in a game of *For Sale* is six. So in each game one player had to sit out. Unfortunately, in recording the results I just made a list of each player's rankings – of course that had the effect of making it impossible to determine who played in each game. Or did it?

Problem Statement

A certain number, p, of players arrive to play a game. Unfortunately, only $p-1$ of them can play at a time. No problem – they just play p rounds with one player sitting out in each round and record their results. For instance with $p = 4$ the results might look like:

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

Each line describes a single game with the underscore denoting the player who sat out. Each column gives all the results for a single player.

Unfortunately, some silly record-keeper forgot to include the underscores and simply collapsed each column, giving:

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

The problem is: how much of the original information can be reconstructed from this data?

For this example it turns out that the complete results can be reconstructed! The first game must clearly have been `_ 2 1 3` or `1 2 _ 3`.

In the former case, the second round results would have to be taken from `1 2 1 3` again (the 1 moved down from the first row for the first player, the rest from the original second row). But the first player can't sit out twice, so it would have to be the third player sitting out in the second round. However, that would lead to two 3's in the final round – something not allowed.

So the first round would have to be `1 2 _ 3`. The second round results now come from `2 2 1 3`, but for the same reason as above, the first player can't sit out in any but the last round. So the second player must sit out the second round, and the fourth player in the third round, giving the original results.

On the other hand, consider this example (with only three players):

```
1 2 2
2 1 1
```

It's easy to check that both the following schedules work:

```
1 2 _
_ 1 2
2 _ 1

1 _ 2
_ 2 1
2 1 _
```

Note here that the results are essentially different - in fact each game in the second version has the opposite result to the game between the same two players in the first version. On the other hand, for this table:

```
1 2 2
1 1 2
```

there are two possible orderings of the games but the results are the same (the first player beats both the second and third players, and the second player beats the third player).

The task below has two parts.

- Construct all the possible results consistent with a given collapsed version.
- Count how many essentially different ones there are (two sets of results that differ only in the order that the games are played are considered the same).

Task

Your program will process an input file from `stdin` and put output to `stdout`. The contents of the input file should be the results of a single series of games between a number of players where each player has a bye in a single round. Correctly formatted input is as above, for example,

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

If there is a failure of formatting (e.g., not all rows have the same number of items, or the number of items in a row is not one more than the number of rows, or there are one or more items that are not integers) then your output should simply be one line:

```
Bad format
```

If the format is correct but the values are not – meaning that some entry or entries are not integers in the range from 1 to the number of rows inclusive, then the output should again be a single line:

```
Bad values
```

Otherwise your program should attempt to construct valid results for the games played. If this is impossible then the output should again be a single line:

```
Inconsistent results
```

Otherwise the output should print all the possible results in grid form as above (using underscore to denote byes) with each set of possible results separated from the next by a blank line.

Finally, in this case, your program should count how many essentially different results there are (where two results must differ by more than the order of games played in order to count as essentially different) and print a final line (separated from the last set of results by a blank line)

```
Different results: <n>
```

Where `<n>` stands for the number of different possible results.

Note that it is usually the case that the best way to determine whether two things differ only in the order of their elements is to find some way to sort their elements and check whether the sorted forms are identical.

(Individual)