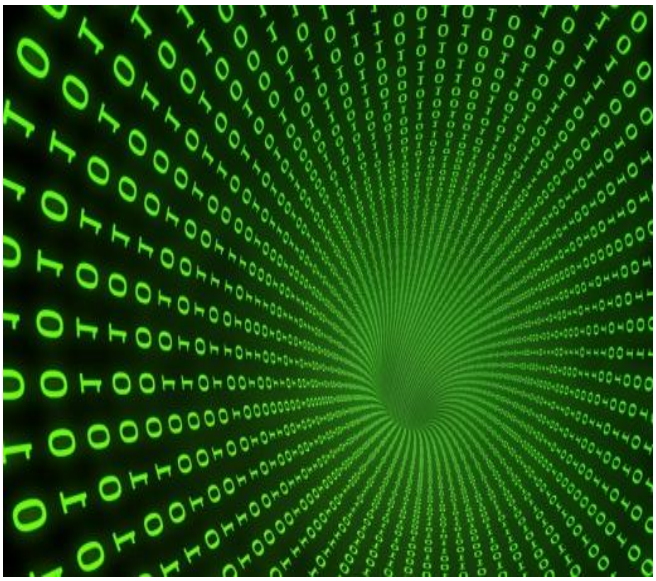
A Binary Matrix

Input: Standard Input
Output: Standard Output



You are given an **M × N** binary matrix. By **M × N** matrix we mean a matrix having **M** rows and **N** columns and by binary matrix we mean each of the **M × N** elements is a binary value, either **0** or **1**. In addition, this matrix wraps both in horizontally and vertically. So $\mathbf{i}^{th}$ row is adjacent to $\mathbf{(i + 1)}^{th}$ row for all **1 ≤ i < M** and $\mathbf{M}^{th}$ row is adjacent to $\mathbf{1}^{st}$ row. Similarly, $\mathbf{i}^{th}$ column is adjacent to $\mathbf{(i + 1)}^{th}$ column for all **1 ≤ i < N** and $\mathbf{N}^{th}$ column is adjacent to $\mathbf{1}^{st}$ column. Obviously row **a** is adjacent to row **b** implies that row **b** is adjacent to row **a**, and same thing is true for columns. Now, two cells of this matrix are adjacent if they are in the same row and their columns are adjacent, or they are in the same column and their rows are adjacent. So for a **3 × 5** matrix, cell **(2, 3)** has 4 adjacent cells **(1, 3), (2, 2), (2, 4), (3, 3)** and cell **(3, 5)** has **4** adjacent cells **(2, 5), (3, 4), (3, 1), (1, 5)**. Note that, by cell **(i, j)** we mean the cell of $\mathbf{i}^{th}$ row and $\mathbf{j}^{th}$ column.

You are only allowed to swap the values of any two adjacent cells of the matrix. Your task is to transform the matrix in such a way so that, each of the rows has same number of **1**s and each of the columns has same number of **1**s. If it is possible print **"both"** and also print the minimum number of swaps required. If it is not possible try to make every row has equal number of 1s. If it is possible print **"row"** and also print the minimum number of swaps required. If it is also not possible try to make every column has equal number of 1s. If it is possible print **"column"** and also print the minimum number of swaps required. If none of these possible you have to print **"impossible"**.

Input

The input starts with an integer **T (T ≤ 10)**, number of test cases.

Each case starts with two integers **M** and **N (2 ≤ M, N ≤ 1000)**, number of rows and columns of the matrix. Next **M** lines denotes **M** rows of the matrix. $\mathbf{j}^{th}$ character of the $\mathbf{i}^{th}$ line denotes the value of cell **(i, j)** of the matrix.

Output

For each case, output a single line. If task is impossible to complete, output **"Case #: impossible"** otherwise print **"Case #: solution_type min_swap"** without quotes, here **#** will be replaced by the case number, **solution_type** will be replaced by the type of solution found as described above it will be one of these three **"both"**, **"row"**, **"column"** without quotes and **min_swap** will be replaced by the minimum number of swaps required to complete the task. Please note that value of **min_swap** can be zero.

See the sample input and output for exact format.

Warning: Input file is large, so use fast input/output, for example instead of using cin/cout use scanf/printf.

Sample Input	Output for Sample Input
2 2 3 001 111 3 3 001 011 000	Case 1: row 1 Case 2: both 2

Explanation of sample input and output:

Case 1

The initial matrix is:

001
111

If we swap values of cell (1, 1) and cell (2, 1), the matrix will become

101
011

Now each row has two 1s and we found the solution.

Case 2

The initial matrix is:

001
01**1**
000

If we swap the values of cell (2, 1) and cell (2, 3) (Considering the matrix wraps), the matrix will become

001
11**0**
000

If we swap the values of cell (2, 1) and cell (3, 1), the matrix will become.

001
010
100

Now each row has one 1 and each column has one 1 and we got our solution.