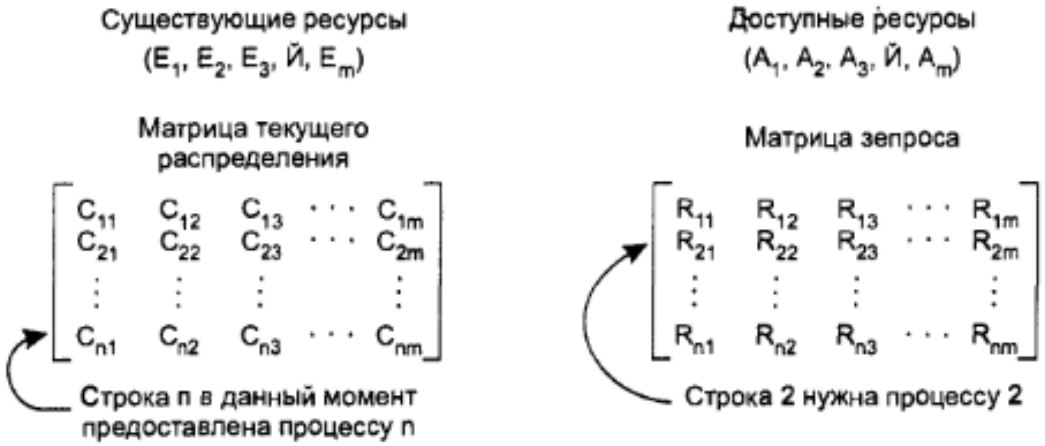
Description

Когда в системе существует несколько экземпляров некоторых из ресурсов, для обнаружения взаимоблокировок необходим другой метод. Сейчас мы расскажем об основанном на матрицах алгоритме, обнаруживающем тупики среди n процессов, от P_1 до P_n. Пусть m — это число классов ресурсов, причем в системе E_1 ресурсов класса 1, E_2 ресурсов класса 2 и, в общем, E_i ресурсов класса i (где $1 \le i \le m$). E — это **вектор существующих ресурсов**. Он передает общее количество имеющихся в наличии экземпляров каждого ресурса. Например, если класс 1 представляет собой накопители на магнитных лентах, то $E_1 = 2$ означает, что в системе есть два магнитофона.

В любой момент времени некоторые из ресурсов могут оказаться занятыми и, соответственно, недоступны. Пусть A будет **вектором доступных ресурсов**, где A_i равно количеству экземпляров ресурса i, доступных в текущий момент (то есть не использующихся). Если оба накопителя на магнитной ленте заняты, A_1 будет равно 0.

Теперь нам нужны два массива: C — **матрица текущего распределения** и R — **матрица запросов**. i-я строка в матрице C говорит о том, сколько представителей каждого класса ресурсов в данный момент использует процесс P_i. Таким образом, C_{ij} — это количество экземпляров ресурса j, которое занимает процесс i. Аналогично, R_{ij} — это количество экземпляров ресурса j, которые хочет получить процесс P_i. Эти четыре структуры показаны на рис. 3.4.



$$\sum_{i=1}^{n} C_{ij} + A_j = E_j.$$

Поочередное выполнение процессов, использующих ресурсы

Имеется множество запущенных на выполнение процессов, при этом во время запуска процессам выделено какое-то количество ресурсов, что зафиксировано в матрице текущего распределения. Процессы обрабатываются в порядке очереди. Если процесс не может выполниться из-за нехватки ресурсов с учетом доступных ресурсов, то управление переходит следующему за ним процессу. После последнего управление переходит на первый процесс.

Если процессу выделено недостаточное количество ресурсов (отражено в матрице запроса), то он может получить недостающие ресурсы только из имеющихся свободных ресурсов. Ресурсы, уже выделенные процессам, не забираются до конца выполнения процессов. После выполнения процесс высвобождает занятые ресурсы.

Требуется выяснить, можно ли в какой-нибудь очередности выполнить все запущенные процессы.

Input

Первая строка входного файла содержит два целых числа разделенные пробелом $0 \le M,N \le 1000$, где M – количество классов ресурсов. N – количество процессов. Далее следует M целых чисел разделенных пробелами, которые представляют собой вектор количества экземпляров каждого класса ресурсов (вектор существующих ресурсов). Далее следуют N строк, где каждая строка соответствует одному процессу, и представляет собой M целых чисел, разделенных пробелами, число есть количество экземпляров каждого класса ресурсов, которое занимает данный процесс (т.е. матрица текущего распределения).
Далее следуют N строк матрицы запроса, где каждая строка соответствует одному процессу, и представляет собой M целых чисел, разделенных пробелами. Здесь соответствующее число есть количество экземпляров данного класса ресурсов, которое нужно добавить помимо уже выделенных данному процессу для его завершения.
Входной файл не содержит лишних пробелов.

Output

Выходной файл содержит строку целых чисел, разделенных пробелами, представляющих номера процессов, которые не могут выполниться (блокируют друг друга). Если система может выполнить все процессы, то выход должен содержать строку “null”

Example

Test.in	Test.out
4 3 4 2 3 1 0 0 1 0 2 0 0 1 0 1 2 0 2 0 0 1 1 0 1 0 2 1 0 0	null
4 3 4 2 3 1 0 0 1 0 2 0 0 1 0 1 2 0 2 0 0 1 1 0 1 0 2 1 0 1	1 2 3