Перебор с возвратом.

Общая схема рекурсивного перебора с возвратом такая:

```
def Перебор(Ситуация):
    if Ситуация конечная:
        Завершающая обработка
    else:
        for Действие in Множество всех возможных действий:
            Применить Действие к Ситуация
            Перебор(Ситуация)
            откатить Действие назад
```

В общем случае алгоритм перебора с возвратом содержит тест, который по данной подзадаче быстро выдаёт один из трёх ответов:

- неудача: подзадача не имеет решения;
- успех: найдено решение подзадачи;
- неопределённость.

Например в задаче о мирных ферзях неудача — новый ферзь бьёт одного из предыдущих, успех — ферзь успешно поставлен на последнюю линию, неопределённость — новый ферзь не бьёт ни одного из старых.
Нерекурсивный общий вид алгоритма перебора обычно удобно организовывать при помощи очереди задач:

```
начать с некоторой задачи P[0]
S = {P[0]} # {очередь активных подзадач}
пока S не пусто:
    выбрать подзадачу P из S и удалить её из S
    разбить P на меньшие подзадачи P[1], P[2], ..., P[k]
    (или сделать k возможных следующих шагов)
    для каждой P[i]:
        если тест(P[i]) = успех:
            вернуть найденное решение
        если тест(P[i]) = неудача:
            отбросить P[i]
        иначе:
            добавить P[i] в S
сообщить, что решения нет
```

Расстановки ферзей

Известно, что на шахматной доске размером 8×8 можно расставить 8 ферзей не бьющих друг друга, причём сделать это можно 92 способами.

Дано натуральное $N \leqslant 12$. Определите все способы расстановки N мирных ферзей на доске $N \times N$.

Требуется вывести количество расстановок и сами расстановки.

В этой задаче надо придумать способ проверки клетки на «занятость» за константное время (не зависящее от размера доски).

Пример вывода см. ниже.

Input	Output
8	92

```
2

. X . .
. . . X
X . . .
. . X .

. . X .
X . . .
. . . X
. X . .
```