Задача A. Киста

Имя входного файла:	`taxi.in`
Имя выходного файла:	`taxi.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Управлять службой такси — сосвем не простое дело. Помимо естественной необходимости централизованного управления машинами для того, чтобы обслуживать заказы по мере их поступления и как можно быстрее, нужно также планировать поездки для обслуживания тех клиентов, которые сделали заказы заранее.

В вашем распоряжении находится список заказов такси на следующий день. Вам необходимо минимимизировать число машин такси, необходимых чтобы выполнить все заказы.

Для простоты будем считать, что план города представляет собой квадратную решетку. Адрес в городе будем обозначать парой целых чисел: x-координатой и y-координатой. Время, необходимое для того, чтобы добраться из точки с адресом (a, b) в точку (c, d), равно $|a - c| + |b - d|$ минут. Машина такси может выполнить очередной заказ, либо если это первый ее заказ за день, либо она успевает приехать в начальную точку из предыдущей конечной хотя бы за минуту до указанного срока. Обратите внимание, что выполнение некоторых заказов может окончиться после полуночи.

Формат входного файла

В первой строке входного файла записано число заказов M ($0 < M < 500$). Последующие M строк описывают сами заказы, по одному в строке. Про каждый заказ указано время отправления в формате `hh:mm` (в интервале с `00:00` по `23:59`), координаты (a, b) точки отправления и координаты (c, d) точки назначения. Все координаты во входном файле неотрицательные и не превосходят 200. Заказы записаны упорядоченными по времени отправления.

Формат выходного файла

В выходной файл выведите единственное целое число — минимальное количество машин такси, необходимых для обслуживания всех заказов.

Пример

taxi.in	taxi.out
2 08:00 10 11 9 16 08:07 9 16 10 11	1
2 08:00 10 11 9 16 08:06 9 16 10 11	2

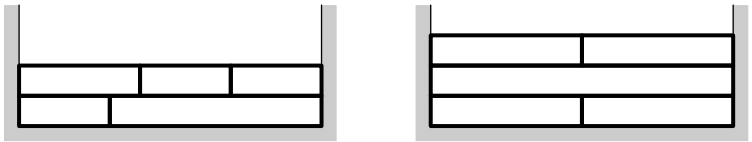
Задача B. Еле Света ясна

Имя входного файла:	`wall.in`
Имя выходного файла:	`wall.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Авторы игры "Веселый тетрис" решили создать новую игру, с названием "Еле Света ясна". Цель этой игры — построить стену чтобы заблокировать проход. Стена должна как можно выше.

Ширина прохода — w. Стена должна быть построена из нескольких кирпичей, которые появляются и падают один за другим. Для каждого кирпича известна его левая и правая граница. Игроку не разрешено перемещать кирпичи, но он может выбрать порядок в котором они падают. Он также может остановить падение кирпичей и сказать, что стена готова.

Стена состоит из нескольких рядов, ширина каждого ряда — w. Также, стена должна быть сплошная, то есть никакие два кирпича не должны иметь одинаковую левую границу, за исключением левой стороны прохода, и никакие два кирпича не должны иметь одинаковую правую границу, за исключением правой стороны прохода. Например на левом рисунке стена — сплошная, а на правом — нет.



Дано описание кирпичей, найдите максимально возможную высоту сплошной стены, которую можно из них составить, а также какие кирпичи надо для этого использовать.

Формат входного файла

Первая строка входного файла содержит два числа n и w — число кирпичей и ширина прохода ($1 \le n \le 2000, 1 \le w \le 2 \cdot 10^9$). Следующие n строк содержат описание кирпичей. Каждый кирпич описывается двумя числами: l_i и r_i — позицией левой и правой границы ($0 \le l_i < r_i \le w$).

Формат выходного файла

На первой строке выходного файла выведите число h — максимальную возможную высоту сплошной стены. Следующие h строк должны описывать ряды. Каждое описание состоит из количества кирпичей в этом ряду, и затем списка номеров этих кирпичей по порядку, справа налево.

Примеры

wall.in	wall.out
6 10 0 3 3 7 7 10 3 10 0 4 4 7	2 1 4 5 6 3
5 10 0 5 0 5 0 10 5 10 5 10	2 1 4 3

Задача C. А-а! пир — баян, раз иены

Имя входного файла:	`pairings.in`
Имя выходного файла:	`pairings.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Разбиением на пары множества чисел $\{1, 2, \ldots, 2n\}$ назовём набор из n пар чисел, в котором каждое число от 1 до $2n$, включительно, встречается ровно один раз. Записывается разбиение на пары так: сначала числа внутри каждой пары сортируются по возрастанию, затем сами пары сортируются по возрастанию первых чисел, и наконец, пары выписываются в строчку по порядку. Разбиения считаются одинаковыми, если они имеют одинаковую запись.

Так, например, для $n = 2$ существует всего три различных разбиения множества чисел $\{1, 2, 3, 4\}$ на пары. Они записываются так:

```
(1 2) (3 4)
(1 3) (2 4)
(1 4) (2 3)
```

Вася и Петя поспорили, кто из них быстрее выпишет все разбиения на пары множества чисел $\{1, 2, \ldots, 2n\}$. Оба одновременно берут по листку бумаги и начинают выписывать все разбиения на пары в лексикографическом порядке. Одно разбиение идёт в этом порядке раньше другого, если для какого-то k от 0 до $2n-1$, включительно, первые k чисел в их записи совпадают, а $(k+1)$-е число первого разбиения меньше, чем $(k+1)$-е число второго.

Оказалось, что Вася выписывает одно разбиение ровно две секунды, а Петя — ровно одну секунду. Вася хочет знать, какое разбиение B Петя выписывал в ту секунду, когда сам Вася заканчивал выписывать некоторое разбиение A. Напишите программу, которая выяснит это для него, чтобы не отвлекать Васю от процесса выписывания.

Формат входного файла

В первой строке входного файла задано целое число n ($1 \le n \le 17$). Во второй строке записаны $2n$ целых чисел через пробел — разбиение A, выписанное Васей. Скобки в записи разбиения опущены для удобства чтения.

Формат выходного файла

Если Петя уже выписал все разбиения до той секунды, в которую Вася закончил выписывать разбиение A, выведите в первой строке выходного файла фразу "`Already finished!`" без кавычек. В противном случае выведите в первой строке выходного файла $2n$ целых чисел через пробел — запись разбиения B, выписанного Петей, также без скобок.

Программы, всегда выводящие "`Already finished!`", будут оценены в 0 баллов.

Примеры

pairings.in	pairings.out
2 1 2 3 4	1 3 2 4
2 1 3 2 4	Already finished!