BOOKING

Pierre is in great trouble today! He is responsible for managing the bookings for the ACM (Acco- modation with Moderate Costs) hotel and has just realized that the booking software has a severe bug. It has created overlapping and wrong room assignments. Pierre is now worried that the hotel might be overbooked. Since the software manufacturer is neither very responsive nor competent, he must check this himself, so that he can timely take countermeasures if necessary.

Fortunately, Pierre was able to export all original bookings (including reservation codes plus correct and valid dates of arrival and departure). The only information that got lost is the time of the booking placements, so that Pierre cannot retrieve any booking priorities (e.g., first-come-first-serve). Using the available information, could you please help Pierre out and tell him a room assignment with the minimum number of rooms that satisfies all bookings? Please be aware that a room must always be cleaned before reuse. Since Pierre does not want to take any risks, he tells you to only consider the maximum cleaning time.

Input

The input consists of several lines. The first line contains $1 \leq T \leq 100$, the number of test cases. Each test case starts with a line containing two integers $1 \leq B \leq 5\,000$, the number of bookings, and $0 \leq C \leq 360$, the cleaning time (in minutes) for one room.

Then follow B lines, each containing the reservation code (a random alphanumeric string of up 20 characters) and the dates of arrival and departure of one booking. Dates are given as strings in the format "YYYY-MM-DD HH:MM" (see example test case), where reservations are only for the years 2013 until 2016.

Output

For each test case, print the minimum number of needed rooms on a single line without any additional blanks. Be aware of leap years but ignore daylight saving time.

Examples

№	stdin	stdout
1	4 2 120 1 2013-07-01 15:59 2013-07-08 16:30 2 2013-07-08 17:30 2013-07-15 12:00 3 60 65 2013-07-08 14:30 2013-07-08 16:00 32 2013-07-01 16:00 2013-07-15 12:00 91 2013-07-01 16:00 2013-07-08 15:00	2 3 1 1

	2 360 a7 2016-02-21 14:00 2016-02-28 21:00 xx 2016-03-01 01:00 2016-03-02 12:57 2 60 a9 2016-02-21 14:00 2016-02-28 11:00 a8 2016-02-28 12:00 2016-03-11 21:00	