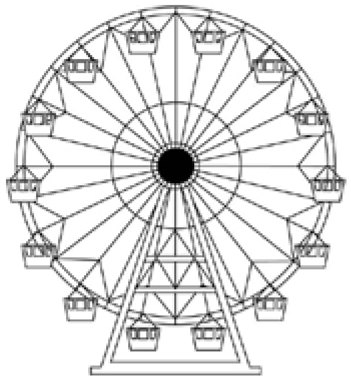
1303 – Ferris Wheel

As we all know that not only children but Ferris-Wheel (FW) is favorite to all ages. That's why the LightOJ Park has made a special FW for the programmers. And unlike other Ferris-Wheels, this FW has **m** seats and every seat is different and very special. Since its circular, the seats are numbered from **1** to **m** in anticlockwise order, so seat 2 is adjacent to seat 1 and seat 3, seat 1 is adjacent to seat 2 and seat **m**.



One day **n** programmers (identified by **1** to **n**) came and each of them wanted to ride in the FW. Since every seat is special, everyone wants to taste the ride of every seat. So, they made the following algorithm:

1. They will form a queue from **1 (front)** to **n (back)**. As they want to enjoy the ride, they want to sit alone in a seat, because two (or more) may start problem-solving in the ride.
2. The FW moves in clockwise order. Initially the seat 1 is in bottom and after 5 seconds, it starts. And when it starts, in each 5 second interval the next seat comes to bottom. So, the order of the seats in bottom are seat 1 (at time 5), seat 2 (at time 10), ..., seat **m** (at time **5*m**), seat 1 (at time **5*m+5**), ... and in this interval, a person sits in that seat or gets out (if he is in the seat) or both (one gets out and another gets in). Assume that these kinds of movements take quite a small time and thus that can be ignored.
3. When a programmer gets out from one seat (he just rode in that seat), then if he has ridden in all the seats, he will leave, otherwise he will join in the back of the queue.
4. When a seat comes into bottom, if all the programmers in the queue have ridden in the seat, nothing happens. Otherwise the first person (from front) in the queue who hasn't ridden in the seat sits in that seat and other programmers keep standing. For example, the current queue is 5, 2, 3, 1 and a seat is in the bottom which has been already ridden by 5 and 2 but 3 hasn't; so, 3 will sit in that seat leaving the queue: 5, 2, 1.

Now you are the **(n+1)**th programmer and you are unlucky, because you are assigned a task, and the task is to find the minimum time when you are sure that everyone has ridden in all the seats.

Input

Input starts with an integer **T (≤ 400)**, denoting the number of test cases.

Each case starts with a line containing two integers **n (1 ≤ n ≤ 20)** and **m (2 ≤ m ≤ 20)**.

Output

For each case, print the case number and the time.

Sample Input	Output for Sample Input

2
2 3
3 2
	Case 1: 65
	Case 2: 40