Task Manager

One day, John was curious about his laptop and wondered how the computer handles many different tasks at the same time. After reading some books, he figured out that the CPU follows first-come-first-serve policy and uses queue to handle different tasks.

In this model, each task has an arriving time **R**, a required processing time **W**, and priority level **P** (high priority or low priority).

CPU will maintain two queues, one queue contains tasks of high priority and the other contains tasks of low priority.

When a new task arrives, the CPU will put it into the queue according to the priority. And when the CPU finishes the current task, it will execute the next task extracted from the high priority queue and from the low priority queue if the high priority queue is empty.

Can you write a program to compute the finishing time of each task?

Input

The input contains multiple test cases. For each test case, the first line contains an integer **n**, indicating that there are **n** tasks.

The following **n** ($1 \leqslant n \leqslant 100$) lines give the task information, where the i-th line contains three integers **Ri**, **Wi**, **Pi**, indicating the information of the i-th task, where $0 \leqslant Ri \leqslant 1000$, $1 \leqslant Wi \leqslant 100$, $0 \leqslant Pi \leqslant 1$ and Pi=0 means high priority.

It is guaranteed that the sequence of arrival times is non-decreasing (if two tasks have the same arrival time and same priority, the task who is earlier in the input is considered to have arrived earlier).

Output

For each test case, print a single line containing the finishing time of each task (Separate the integers by a space and no more space after the last integer in the line).

Sample Input	Sample Output
3 0 4 1 0 2 0 3 4 0 3 0 4 0 0 2 0 3 4 0 5 3 4 1 5 3 0 5 7 1 6 3 0 6 6 0 5 3 4 1 5 3 0 5 7 1 14 3 0 26 6 0	6 2 10 4 6 10 7 10 26 13 19 7 10 17 20 32

Explanation:

In the first sample, at time 0, both task 1 and task 2 arrives and they are put into different queues. The machine executes task 2 first as task 2 is in the high priority queue. At time 2, task 2 finishes and the machine executes task 1 from low priority queue. At time 3, task 3 arrives and it is in high priority queue. At time 6, task 1 finishes and the machine executes task 3 from high priority queue.