Problem G. Restoring the Duration of Tasks

Time limit	2000 ms
Mem limit	262144 kB

Recently, Polycarp completed n successive tasks.

For each completed task, the time s_i is known when it was given, no two tasks were given at the same time. Also given is the time f_i when the task was completed. For each task, there is an unknown value d_i ($d_i > 0$) — **duration of task execution**.

It is known that the tasks were completed in the order in which they came. Polycarp performed the tasks as follows:

- As soon as the very first task came, Polycarp immediately began to carry it out.
- If a new task arrived before Polycarp finished the previous one, he put the new task at the end of the queue.
- When Polycarp finished executing the next task and the queue was not empty, he **immediately** took a new task from the head of the queue (if the queue is empty — he just waited for the next task).

Find d_i (duration) of each task.

Input

The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases.

The descriptions of the input data sets follow.

The first line of each test case contains one integer n ($1 \le n \le 2 \cdot 10^5$).

The second line of each test case contains exactly n integers $s_1 < s_2 < \dots < s_n$ ($0 \le s_i \le 10^9$).

The third line of each test case contains exactly n integers $f_1 < f_2 < \dots < f_n$ ($s_i < f_i \le 10^9$).

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each of t test cases print n positive integers $d_1, d_2, \dots, d_n$ — the duration of each task.

Examples

Input	Output
`4`	`2 7 1`
`3`	`1 1`
`0 3 7`	`1 183 1 60 7644 900 914 80152 5644467`
`2 10 11`	`1000000000`
`2`	
`10 15`	
`11 16`	
`9`	
`12 16 90 195 1456 1569 3001 5237 19275`	
`13 199 200 260 9100 10000 10914 91066 5735`	
`1`	
`0`	
`1000000000`	

Note

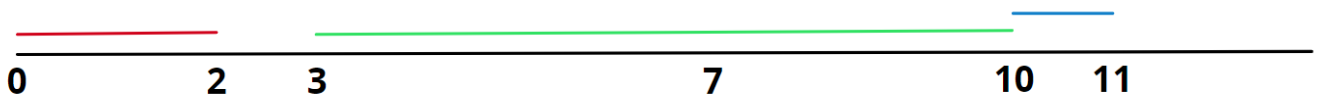
First test case:

The queue is empty at the beginning: $[\,]$. And that's where the first task comes in. At time 2, Polycarp finishes doing the first task, so the duration of the first task is 2. The queue is empty so Polycarp is just waiting.

At time 3, the second task arrives. And at time 7, the third task arrives, and now the queue looks like this: $[7]$.

At the time 10, Polycarp finishes doing the second task, as a result, the duration of the second task is 7.

And at time 10, Polycarp immediately starts doing the third task and finishes at time 11. As a result, the duration of the third task is 1.



An example of the first test case.