Network packet processing simulation

Problem Introduction

In this problem you will implement a program to simulate the processing of network packets.

Problem Description

Task. You are given a series of incoming network packets, and your task is to simulate their processing. Packets arrive in some order. For each packet number 𝑖, you know the time when it arrived 𝐴𝑖 and the time it takes the processor to process it 𝑃𝑖 (both in milliseconds). There is only one processor, and it processes the incoming packets in the order of their arrival. If the processor started to process some packet, it doesn’t interrupt or stop until it finishes the processing of this packet, and the processing of packet 𝑖 takes exactly 𝑃𝑖 milliseconds.

The computer processing the packets has a network buffer of fixed size 𝑆. When packets arrive, they are stored in the buffer before being processed. However, if the buffer is full when a packet arrives (there are 𝑆 packets which have arrived before this packet, and the computer hasn’t finished processing any of them), it is dropped and won’t be processed at all. If several packets arrive at the same time, they are first all stored in the buffer (some of them may be dropped because of that — those which are described later in the input). The computer processes the packets in the order of their arrival, and it starts processing the next available packet from the buffer as soon as it finishes processing the previous one. If at some point the computer is not busy, and there are no packets in the buffer, the computer just waits for the next packet to arrive. Note that a packet leaves the buffer and frees the space in the buffer as soon as the computer finishes processing it.

Input Format. The first line of the input contains the size 𝑆 of the buffer and the number 𝑛 of incoming network packets. Each of the next 𝑛 lines contains two numbers. 𝑖-th line contains the time of arrival 𝐴𝑖 and the processing time 𝑃𝑖 (both in milliseconds) of the 𝑖-th packet. It is guaranteed that the sequence of arrival times is non-decreasing (however, it can contain the exact same times of arrival in milliseconds — in this case the packet which is earlier in the input is considered to have arrived earlier).

Constraints. All the numbers in the input are integers. 1 ≤ 𝑆 ≤ 105; 1 ≤ 𝑛 ≤ 105; 0 ≤ 𝐴𝑖 ≤ 106; 0≤𝑃𝑖 ≤103;𝐴𝑖 ≤𝐴𝑖+1 for1≤𝑖≤𝑛−1.

Output Format. For each packet output either the moment of time (in milliseconds) when the processor began processing it or −1 if the packet was dropped (output the answers for the packets in the same order as the packets are given in the input).

Time Limits. C: 2 sec, C++: 2 sec, Java: 6 sec, Python: 8 sec. sec, Ruby: 6 sec, Scala: 6 sec.

Memory Limit. 512Mb.

Sample 1.

Input:

1 0

Output: 

Explanation: If there are no packets, you shouldn’t output anything.

Sample 2.

Input:

1 1

0 0

Output:

0

Explanation: The only packet arrived at time 0, and computer started processing it immediately.

Sample 3.

Input:

1 2

0 1

0 1

Output:

0

-1

Explanation: The first packet arrived at time 0, the second packet also arrived at time 0, but was dropped, because the network buffer has size 1 and it was full with the first packet already. The first packet started processing at time 0, and the second packet was not processed at all.

Sample 4.

Input:

1 2

0 1

1 1

Output:

0

1

Explanation: The first packet arrived at time 0, the computer started processing it immediately and finished at time 1. The second packet arrived at time 1, and the computer started processing it immediately.