Problem 2. Video buffering

Input file:	`input.txt`
Output file:	`output.txt`
Time limit:	1 second
Memory limit:	256 megabytes

You are given a video in the MPEG-2 format. For each frame of a video, it is known how much time it takes to decode it. Find the minimal necessary buffer size, such that the whole video could be demonstrated without lags and skipped frames.

The video file contains a sequence of frames in the order they must be shown on the screen. The video must be shown at constant frame rate: the time interval d between two adjacent frames is specified in the file. If we number the frames from zero up, the 0-th frame must be shown at the moment of time 0, the first frame at time d, the second frame at time $2d$, the third frame at time $3d$, etc.

For a video frame to be shown on the screen it has to be decoded first. All frames which have been decoded and are being decoded must be stored in the buffer, except for the frames which will never be needed anymore for sure. The buffer size is set in frames and remains fixed throughout the demonstration of the video. Usually, a frame can be discarded from the buffer once it has been shown, however, this is not always the case, because sometimes in order to decode one frame, you have to have some other frames in the buffer in decoded state. To begin decoding frame f, it is necessary to:

1. have all frames, upon which the frame f depends, decoded in the buffer, and
2. have free slot in the buffer, where the frame is placed for the period of decoding.

Decoding time is specified for each individual frame. No more than one frame can be decoded at any given moment. A decoded frame f can be discarded from the buffer only if:

1. the frame f has already been shown, and
2. all frames dependent on the frame f have already been decoded.

There are three types of frames in the video file:

- `I`-frame. This frame does not depend on anything.
- `P`-frame. Depends on the preceding frame of the type `I` or `P` (the nearest one).
- `B`-frame. Depends on the preceding frame of the type `I` or `P`, and on the following frame of the type `I` or `P` (both frames being the nearest).

For instance, if there are five frames with the types `IBBBP`, the first frame is completely independent, the last frame depends on the first frame, and the remaining frames depend on both the first and the last frame.

The decoding sequence is strictly predefined. Frames must be decoded in the same order as they are to be shown on the screen. The only exception is when a `B`-frame is encountered, which depends on a later frame. In this case the later frame must be decoded first, if it has not been decoded yet. For instance, if frames of the types `IBBPBBI` are given, they must be decoded in the following order: 0, 3, 1, 2, 6, 4, 5.

Find the minimal buffer size required to show all frames of the video strictly on time. It is allowed to begin the decoding any time before the beginning of the video demonstration. Assume that showing a frame on screen occurs instantaneously and takes no time, as well as placing and discarding a frame in the buffer.

Input

The first line of the input file contains two integers: N — the number of frames, d — the time between adjacent frames in microseconds ($3 \le N \le 2 \cdot 10^5$, $1 \le d \le 10^9$).

The remaining N lines describe the frames in the order they must be shown. Each frame is defined by its type (`I`, `P` or `B`) and the amount of time in microseconds required to decode it. The decoding time is an integer in the range from 1 to 10^9 inclusive.

It is guaranteed that the first frame has the type `I`, and the last frame has the type `I` or `P`.

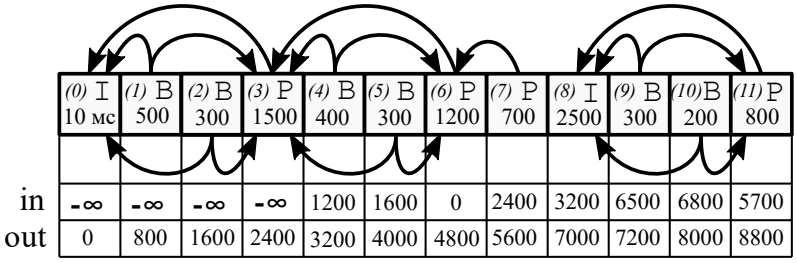
Output

The output file must contain a single integer: the minimal size of the buffer in frames, such that all video is demonstrated without lags.

Example

input.txt	output.txt
12 800 I 10000 B 500 B 300 P 1500 B 400 B 300 P 1200 P 700 I 2500 B 300 B 200 P 800	4

Example explanation



decoding order: 0, 3, 1, 2, 6, 4, 5, 7, 8, 11, 9, 10;

The illustration shows a method of playing the video using a 4-frame buffer. Arrows show frame dependencies, and the in/out lines show when the frame is placed into the buffer and when it is discarded from the buffer. All frames are numbered from zero up.