Problem C: GPT DarkDown

Mahdieh is a developer working on a chatbot for Divar's open platform (Kenar Divar). She wants to use ChatGPT's API for her chatbot. This means the chatbot receives messages from ChatGPT and forwards them to the user. When the chatbot asks ChatGPT to generate a message, ChatGPT sends the message in chunks to the chatbot. So for each chunk, there is a time when Mahdieh's chatbot receives it, and there is a string it contains.

For a smooth user experience, Mahdieh wants her chatbot to simulate continuous typing, outputting 1 character per millisecond. Obviously, no character can be typed before it is received, so Mahdieh must ensure that the current typing character is available for her chatbot before typing it. To make the perfect smooth user experience she wants to see in her chatbot, the chatbot waits for a while without typing anything, and then starts typing non-stop. Now Mahdieh has all the chunks' information from a hypothetical message, and she needs your help to tell her the first time the chatbot can start to type.

However, ChatGPT (and therefore Mahdieh's chatbot) applies **Darkdown** formatting to the chunks of text, which includes **bold**, *italic*, `inline code`, and even emojis! 😄

So, there are formatting characters in the received message from ChatGPT that will not be rendered in the final text. You need to determine only the (visible) rendered content for smooth typing. For example, the Darkdown text "`[Let's] (Code!) :rocket:`" consists of 24 characters, but its rendered text is only 13 characters long: "*Let's* **Code!** 🚀".

Your task is to determine the earliest time Mahdieh can start typing the rendered output for a smooth experience.

Input

The input starts with a line containing a single integer k, as the number of chunks ($1 \leqslant k \leqslant 10^5$). The next k lines contain the complete input message, where every line represents a non-empty chunk. You can assume the ChatGPT message does not contain newline characters, and the chunks could start or end with spaces. It is guaranteed that the total number of characters in all the chunks combined will not exceed 10^5, and the final rendered message will not be empty. The last line contains k space-separated integers, $t_1, t_2, \ldots, t_k$, where t_i is the time the chatbot receives the i^{th} chunk from ChatGPT ($1 \leqslant t_1 < t_2 < \cdots < t_k \leqslant 10^9$).

The ChatGPT message has the following **Darkdown** formatting:

- **Inline Code Blocks**:
  - Inline code is given inside a pair of backtick characters, e.g. `` `code` ``. The backtick charcters themselves are not rendered in the final output. You can assume there is no backtick characters inside the inline code.
  - Formatting markers inside inline code blocks (like `(` or `\`) are rendered as literal characters and are not parsed as formatting.
- **Special Characters**:
  - In order to render the special characters (`(`, `)`, `[`, `]`, `:`, `\`, `` ` ``), it is sufficient to escape them by preceding them with a backslash character (e.g. `\(`, `\)`, `\[\]`, `\]`, `\:`, `\\`, `` \` ``).

  - Double backslashes (\\) are rendered as a single literal backslash in the visible output.
  - It is guaranteed that every backslash in the input is followed by a special character, except in inline codes and after an escaped backslash (\\).
- **Bold and Italic Formatting**:
  - Bold formatting can be applied by enclosing the text with parentheses: `(bold)`.
  - Italic formatting can be applied by enclosing the text with square brackets: `[italic]`.
  - It is not permitted to have nested blocks of bold or italic text. Also, a block of text cannot be simultaneously bold and italic. For example, it is not allowed to have `[(bold and italic) italic]` or `((superbold) bold)`.
  - Inline codes or emojis are allowed inside a bold or italic text.
- **Emojis**:
  - An emoji is given as a non-empty string of lowercase English letters enclosed within a pair of colons (e.g. `:smile:`, `:rocket:`).
- **Punctuation and Spaces**:
  - All standard punctuation marks (`.`, `,`, `!`, `?`, `'`, `-`, `/`) and space characters are rendered normally.

It is guaranteed that the input adheres strictly to the Darkdown formatting rules described above, and the final rendered text is unique.

In order to find the answer, you have to consider only the visible (rendered) characters, which include:

- **Plain text**: All letters, numbers, spaces, escaped special characters, and standard punctuation marks not part of any formatting.
- **Emojis**: Represented as a single conceptual character. For example, `:smile:` is rendered as 😄. Note that the conceptual character for an emoji is ready for typing when the chatbot has received its ending colon (`:`).
- **Inline Code**: The content inside backtick characters.

Output

Output the earliest time Mahdieh's chatbot can start typing the message such that the chatbot prints the whole text smoothly.

Example

Standard Input	Standard Output
11 The (International Collegiate Programming Contest) \(`ICPC`\) is a global [competi tive programming contest] focusing on algorithm ic problem-solving and teamwork. [Le t's] test `some` (more) Darkdown (elemen ts) here. Visit the Of ficial ICPC Website\: \(https\://i cpc.global\) for more info! :rocket: 10 20 30 40 50 60 70 80 90 100 110	10
2 l ML5g(RsXFVBdO R)(`gwkz MV `) 10 1000	10

Note

For better understanding, these are the final visible rendered messages of the example inputs:

Sample 1:
The **International Collegiate Programming Contest** (ICPC) is a global *competitive programming contest* focusing on algorithmic problem-solving and teamwork. *Let's* test `some` **more** Darkdown *elements* here. Visit the Official ICPC Website: (https://icpc.global) for more info! 🚀

Sample 2:
l ML5g **RsXFVBdO R**`gwkz MV`