Structuring the Document

A document is represented as a collection paragraphs, a paragraph is represented as a collection of sentences, a sentence is represented as a collection of words and a word is represented as a collection of lower-case ([a-z]) and upper-case ([A-Z]) English characters. You will convert a raw text document into its component paragraphs, sentences and words. To test your results, queries will ask you to return a specific paragraph, sentence or word as described below.

Alicia is studying the C programming language at the University of Dunkirk and she represents the words, sentences, paragraphs, and documents using pointers:

- A word is described by:

```
struct word {
    char* data;
};
```

- A sentence is described by:

```
struct sentence {
    struct word* data;
    int word_count;//the number of words in a sentence
};
```

The words in the sentence are separated by one space (" "). The last word does not end with a space.

- A paragraph is described by:

```
struct paragraph {
    struct sentence* data  ;
    int sentence_count;//the number of sentences in a paragraph
};
```

The sentences in the paragraph are separated by one period (".").

- A document is described by:

```
struct document {
    struct paragraph* data;
    int paragraph_count;//the number of paragraphs in a document
};
```

The paragraphs in the document are separated by one newline("\n"). The last paragraph does not end with a newline.

For example:

Learning C is fun.
Learning pointers is more fun.It is good to have pointers.

- The only sentence in the first paragraph could be represented as:

```
struct sentence first_sentence_in_first_paragraph;
first_sentence_in_first_paragraph.data = {"Learning", "C", "is", "fun"};
```

- The first paragraph itself could be represented as:

```
struct paragraph first_paragraph;
first_paragraph.data = {{"Learning", "C", "is", "fun"}};
```

- The first sentence in the second paragraph could be represented as:

```
struct sentence first_sentence_in_second_paragraph;
first_sentence_in_second_paragraph.data = {"Learning", "pointers", "is", "more", "fun"};
```

- The second sentence in the second paragraph could be represented as:

```
struct sentence second_sentence_in_second_paragraph;
second_sentence_in_second_paragraph.data = {"It", "is", "good", "to", "have", "pointers"};
```

- The second paragraph could be represented as:

```
struct paragraph second_paragraph;
second_paragraph.data = {{"Learning", "pointers", "is", "more", "fun"}, {"It", "is", "good", "to", "have",
"pointers"}};
```

- Finally, the document could be represented as:

```
struct document Doc;
Doc.data = {{{"Learning", "C", "is", "fun"}}, {{"Learning", "pointers", "is", "more", "fun"}, {"It", "is",
"good", "to", "have", "pointers"}}};
```

Alicia has sent a document to her friend Teodora as a string of characters, i.e. represented by **`char*`** not **`struct document`**. Help her convert the document to **`struct document`** form by completing the following functions:

- **`void initialise_document(char* text)`** to intialise the document. You have to intialise the global variable **Doc** of type **`struct document`**.
- **`struct paragraph kth_paragraph(int k)`** to return the k^{th} paragraph in the document.
- **`struct sentence kth_sentence_in_mth_paragraph(int k, int m)`** to return the k^{th} sentence in the m^{th} paragraph.
- **`struct word kth_word_in_mth_sentence_of_nth_paragraph(int k, int m, int n)`** to return the k^{th} word in the m^{th} sentence of the n^{th} paragraph.

Input Format

The first line contains the integer $paragraph\_count$.
Each of the next $paragraph\_count$ lines contains a paragraph as a single string.

The next line contains the integer q, the number of queries.
Each of the next q lines contains a query in one of the following formats:

- $1\ \ k$: This corresponds to calling the function **kth_paragraph**.
- $2\ \ k\ \ m$: This corresponds to calling the function **kth_sentence_in_mth_paragraph**.
- $3\ \ k\ \ m\ \ n$: This corresponds to calling the function **kth_word_in_mth_sentence_of_nth_paragraph**.

Constraints

- The text which is passed to **get_document** has words separated by a spaces(" "), sentences separated by a period(".") and paragraphs separated by a newline("\n").
- The last word in a sentence does not end with a space.
- The last paragraph does not end with a newline.
- The words contain only upper-case and lower-case English letters.
- $1 \leq$ number of characters in the entire document ≤ 1000.
- $1 \leq$ number of paragraphs in the entire document ≤ 5.

Output Format

Print the paragraph, sentence or the word corresponding to the query to check the logic of your code.

Sample Input 0

```
2
Learning C is fun.
Learning pointers is more fun.It is good to have pointers.
3
1 2
2 1 1
3 1 1 1
```

Sample Output 0

```
Learning pointers is more fun.It is good to have pointers.
Learning C is fun
Learning
```

Explanation 0

The first query returns the second paragraph.
The second query returns the first sentence of the first paragraph.
The third query returns the first word of the first sentence of the first paragraph.