CMPUT 274/5 - Tangible Computing
Morning Problem: Decoder

Description

My attic is full of old stuff. The other day I came across a weird book that had only zeros and ones in it, plus a dictionary at the end of the book.

The dictionary lists pairs of strings, in each pair the first string composed of zeros and ones, while the second string is an English word. I tried the following simple technique to figure out what's in the book: Start from the beginning of the zeros and ones. Go until I find a string of zeroes and ones that appears in the dictionary. Write down the corresponding English word. Then continue the same way with the remaining text. This seems to work. However, I got exhausted doing this by hand.

Your job is to write a script **decoder.py** that helps me. You are guaranteed that at each step of the decoding algorithm there is a unique word in the dictionary whose zero/one sequence is a prefix of the part of the text I have yet to decipher.

Input

The first line contains the number of elements in the dictionary, n. The next n lines contain these individual pairs of the dictionary, separated by whitespace. The final line contains the coded text as a sequence of zeroes and ones. The decoding should start at the beginning of the coded text.

The dictionary may have words in it that never appear in the text. The dictionary has at most 1000 pairs in it. The binary string in each pair has length at most 50. The length of the text to be decoded is at most 1000 characters.

Output

Print the words of the decoded text here, separated by spaces, in a single line with a new line at the end.

Sample Input 1

```
3
110 up
101011 picking
101001 and
110101001101011
```

Sample Output 1

```
up and picking
```

Explanation: Starting from the beginning of this string, we see that "1" and "11" are not in the dictionary as keys, but "110" is. Hence, after processing three characters of the input, we decode the word "up" (this is the word next to "110" in the dictionary). Continuing with

the remaining string, "1","10","101","1010","10100" are not in the dictionary, but "101001" is, and corresponds to the word "and", hence this will be the second word of the decoded text. The remaining string is "101011". This is processed in the same way as before and is found to correspond to the word "picking".

Sample Input 2

```
7
011 rabbit
000 ran
10011 with
1111 white
11011 close
10000 pink
11001 eyes
11110111001110000110010001101 1
```

Sample Output 2

```
white rabbit with pink eyes ran close
```