## Problem 4 - Variable Length Coding

Consider the following simple algorithm for encoding (i.e. compressing) text:

* Count the occurrences of each symbol in each symbol in the text
  + i.e. the “frequency” of a symbol in the text
* The most frequent symbol will be encoded with a single bit “1”
* The second most frequent will be encoded with two bits: “11”
* i.e. if we **order the symbols by their frequency**, having the **most frequent symbol at position 1** and the **least frequent symbol at position N**:
  + the **1**st symbol has a code of **one** bit with a value of 1
  + the **2**nd symbol has a code of **two** bits with a value of 1
  + …
  + the **i**th symbol has a code of **i** bits with a value of 1
  + …
  + the **N**th symbol has a code of **N** bits with a value of 1
* Go through the text and **encode each symbol according to its code** from the above rules
  + **Separate encoded symbols by placing a single “0”** bit between them
  + E.g. if the symbol “a” has to be encoded with the code “1” and “b” has to be encoded with the code “11”, then the text “ab” will be encoded as “1011”
* The **encoded text is then padded with zeroes to the right**, so its **total length is divisible by 8** (the size of a byte)
  + E.g. for the previous example, where the entire encoded text was “1011”, it will be padded right with 4 zeroes, giving “10110000”
* The entire encoded text is **split into groups of 8 symbols** and **each of the groups is turned into a 1-byte unsigned integer value** (i.e. “parsed” from binary into a byte-sized integer)
  + For the previous example, where the encoded and padded text was “10110000”, there will be exactly 1 group, so the end result will be a single byte integer: 176
    - The leftmost bit is considered the most-significant
* After all the values for the encoded text, we append the **code table**
  + The code table is a series of lines, each of which a concatenation of a symbol and a number, representing the length of its code
  + E.g. from the previous example the text “ab” will have the following code table:
    - a1  
      b2
    - Note: the code length could be more than 1 digit, e.g. a text with the English alphabet, the last line of the code table will be “z26”
* A broader example: say we have the text “abaaba”.
  + The most frequent symbol is “a”, followed by “b”.
  + “a” will have the code “1” and “b” will have the code “11”
  + The encoded text will be “1011010101101”, which has a length of 13
  + The encoded text will be padded right with 3 zeroes, giving “1011010101101000”
  + “1011010101101000” will be divided into two groups of 8 symbols:
    - “10110101” and
    - “01101000”
  + “10110101” will be turned into the integer 181 and “01101000” will be turned into the integer 104
  + The end result is the entire text (which was 6 character values) encoded in two bytes, ordered like so: 181 104
  + Of course, we append the code table (each symbol with its code) after the text itself

Now, that’s all well and good, but what’s the use of encoded text if you can’t decode it? That’s where you come in.

Your task is to write a program, which **given an encoded by the above rules text** (as series of integers, followed by a series of lines of a code table) **decodes the integers and prints the original text**.

### Input

The input data should be read from the console.

On the **first line** of the input, there will be a **sequence of integers**, separated by **spaces** (representing the encoded **text)**

**On the next** line, there will be a single integer number **L** – the **number of lines** in the **code table**

On **each** of the next **L lines** there will be a **string**.

* The **first symbol** of the string describes **which symbol from the text this line of the table represents**.
* All of the remaining symbols will be **digits**, representing a **number** (most significant digit is the second symbol in the string), equal to the **length of the code for the symbol** in the text

The input data will always be valid and in the format described. There is no need to check it explicitly.

### Output

The output data should be printed on the console. Print exactly one line – the original text (decoded)

### Constraints

* The original text will be no more than 4000 symbols
* Each symbol in the original text will have an ASCII code from 0 to 255

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 251 253 214 255 223 187 254 254 183 175 254 240  11  2  S5  a6  e1  l7  m3  o8  p9  s10  t4  x11 | Some sample text |
| **Input** | **Output** |
| 173 222  4  2  a1  b3  c4 | aa bc |

Note: both examples have a space (" ") character (in both examples it has a code length of 2)