The Huffman Decoder

Huffman codes compress text by assigning the characters that occur at the highest frequency the shortest possible codes. In this encoding scheme, no code can be a prefix of another. For example, if the code for *a* is *01*, then the code for *b cannot* be *011*.

Given an array of Huffman code mappings and a Huffman-encoded string, find the decoded string. Each mapping will be a string consisting of a tab-separated character and its encoded value: *'c   encoded value'* where the whitespace is a tab character. The newline character is represented as the character *[newline]* in the codes list, but should translate to *n* when decoded.

For example, given *codes = ('a 100100','b 100101','[newline] 111111')* and the string *encoded = 100100111111100101* we do the following.
Break *encoded* into its constituent encodings.

```
        100100
        111111
        100101
```

Now map them to their characters and return the string: *'a\nb'*. This will print as:

```
        a
        b
```

Note: While all code mappings in the example are *6* digits long, mappings can be different lengths. The algorithm creates the shortest length mapping for the most frequent character encoded.

Function Description
Complete the function *decode* in the editor below. The function must return the decoded string.

decode has the following parameter(s):
  codes[codes[0],...codes[n-1]]: an array of character mappings
  encoded: an encoded string

Constraints
- *1 ≤ n ≤ 100*
- *1 ≤ |encoded| ≤ 7000*
- *All characters of encoded are either '0' or '1'*