LC BB ND CNDY

Suppose the phrase has n consonants. Then, there will be 2^{n-1} different clues that can be produced this way—for each i such that $1 \leq i < n$, we can either put a space between the ith and $(i+1)$th consonant, or not.

In combinatorics, it's natural to establish encode each such clue into a corresponding bitstring of length n—let its ith character be `0` if there is a space between the ith and $(i+1)$th letters, and `1` if there is none.

In order to find the kth element of $\mathcal{L}$ without literally constructing the entire list, we use the following cute fact:

Claim: Let A and B be clues, and let a and b be their corresponding bitstrings. Then, $A < B$ if and only if $a < b$.

Proof: Recall that the space is a symbol that comes before all the letters of the alphabet, lexicographically. By definition, lexicographic comparisons are made by finding the first character from the left where the two words differ. In this case, at that spot, one will have a space and one won't. The word with the space there comes first in $\mathcal{L}$.

But think of what that means in terms of the bitstrings—if $A < B$, then we examine the leftmost place in their bitstrings where one has a `0` (a space) and the other has a `1` (doesn't have a space). But this corresponds to the normal interpretation of when bitstring a is less than bitstring b. QED.

But the neat thing about bitstrings is that order is preserved even if we interpret them as binary representations of integers! That is, if a and b are bistrings, and x and y are their corresponding integers if they are read as binary notation, then $a < b$ if and only if $x < y$.

But that means the first element of $\mathcal{L}$ is the one corresponding to 0, the second element of $\mathcal{L}$ is the one corresponding to 1, the third element of $\mathcal{L}$ is the one corresponding to 2... and so on. In general, the kth entry in $\mathcal{L}$ is the clue whose bitstring is the binary representation of $k-1$

In summary, we have a remarkably simple $\mathcal{O}(|s|)$ solution.

- Filter out all non-consonants in s, and let n be the number of consonants in the clue.
- If $k > 2^{n-1}$, output `out of bounds`
- If not, read out the $(n-1)$-bit binary representation of $k-1$, from left to right.
- If the ith bit from the left is `0`, then insert a space between the ith and $(i+1)$th consonants.

The model solution uses bitwise operations to extract the binary representation of $k-1$. Feel free to use `%2` and `/2` instead, or perhaps your favorite language's built-in binary converter.