# Approach

Time Complexity:- O(n)

Approach

1. Calculate Total Size:
    - Go via the string s and compute the whole length of the decoded string (size).
    - If you encounter a digit, multiply length by way of that digit's value.
    - Else, increment size by way of 1.
2. Decode in Reverse:
    - Start iterating through s in reverse.
    - At every step, update k by using taking its modulo with the contemporary length.
    - If ok turns into zero and the cutting-edge individual is a letter, go back it.
    - If it is a digit, adjust size through dividing it by the digit's value.
    - If it's a letter, decrease size by means of 1.
3. If the loop completes with out locating the person, go back an empty string. This manner k exceeds the size of the decoded string.
   Example -
   s = "leet2code3" and k = 10:
   First loop to find the length of the string
   Explanation:
   Understood, you'd like the explanation without the code. Here's the explanation:

Given s = "leet2code3" and k = 10:

1. Calculate the size of the decoded string:
    - s[0] = 'l': size += 1 (size becomes 1)

- s[1] = 'e': size += 1 (size becomes 2)
- s[2] = 'e': size += 1 (size becomes 3)
- s[3] = 't': size += 1 (size becomes 4)
- s[4] = '2': size *= 2 (size becomes 8)
- s[5] = 'c': size += 1 (size becomes 9)
- s[6] = 'o': size += 1 (size becomes 10)
- s[7] = 'd': size += 1 (size becomes 11)
- s[8] = 'e': size += 1 (size becomes 12)
- s[9] = '3': size *= 3 (size becomes 36)

The size of the decoded string is now 36.

2. Start decoding from the end of the string:
    - Iteration 1 (i = 9):
        - k = 10, size = 36
    - Iteration 2 (i = 8):
        - k = 10, size = 12
    - Iteration 3 (i = 7):
        - k = 10, size = 11
    - Iteration 4 (i = 6):
        - k = 0, size = 10 (k becomes 0, and "o" is returned)

The character at position 10 in the decoded string is "o".