1. PROBLEM STATEMENT

The optimal solution is to remove the last two elements to reduce x to zero

Given:

Input string: “a2345678999999999999999”

Input K = 1

Output = a

The input string contains both letters and numbers

1. The encoded string is read one character at a time.
2. If the character is a letter, that letter is written on to the tape
3. If character read is a digit, the entire current tape is repeatedly by the digit

Problem Solution

This can be achieved by checking the size of decoded string:

Input string: “a2345678999999999999999”

a. Count the number of the string
b. When I encounter a digit
c. I will multiply the last character to the digit and it continue

Calculate the value of k. I will use the logic below:

k = k%size = 0,

Whenever this condition is meet for any character in the string that character will be the output

2. ALGORITHM

ALGORITHM	Decoded String at Index
INPUT:	String `inputString` & Integer k. `(String inputString, int k)`
OUTPUT:	Result of operation
STEP 1:	**START**

STEP 2:	a. Initialize the counter
	`long size = 0;`
	b. Initialize the size of input string
	`int len = inputString.length();`

STEP 3:	a. Iterate through the characters given on the inputString
	`for(int i = 0; i < len; i++)`
	
	b. Save the index location character in a variable ch
	`char ch = inputString.charAt(i)`
	
	c. Check if the character at index is a digit
	`if(Character is Digit))`
	
	If the condition is True:
	d. Multiply the digit at the index location by the last character digit
	`size = size * Integer.parseInt("" + ch);`
	
	Else:
	e. increment the counter
	`size++;`

STEP 4:	a. Iterate through the characters given on the inputString in reverse direction starting from the last element
	`for(int i = len - 1; i >= 0; i--)`
	
	b. Save the index location character in a variable ch
	`char ch = inputString.charAt(i);`
	
	c. Calculate the modulus of k
	`K = k%size`
	
	d. Check if Character at index is a letter && if modulus of k = 0
	
	`if(Character is letter) && k == 0)`
	
	If the condition is True:

	e. Return that character and convert it to string

STEP 5:	a. Check if character at index location is a digit
	`if(Character is Digit(ch))`
	
	If the condition is True:
	b. Divide the size with the previous character length
	`size = size/Integer.parseInt("" +ch);`
	
	Else:
	
	c. Divide decrement the size by 1
	`size--;`

STEP 6:	Return null
STEP 7:	**STOP**

3. CODE

```java
package com.shedrack.solution;

public class DecodeAtIndex {

	public static void main(String[] args) {
		String inputString = "ha22";
		decodeString(inputString, 5);
	}

	public static String decodeString(String inputString, int k) {

		long size = 0;
		int len = inputString.length();

		//Length of the decoded string - size

		for(int i = 0; i < len; i++) {
			char ch = inputString.charAt(i);
				if(Character.isDigit(ch)) {
					size = size * Integer.parseInt("" + ch);
				}
				else {
					size++;
				}
		}
		for(int i = len - 1; i >= 0; i--) {
			char ch = inputString.charAt(i);
			k%=size;
			if(Character.isLetter(ch) && k == 0) {
				System.out.println(Character.toString(ch));
				return Character.toString(ch);

			}
			if(Character.isDigit(ch))
				size = size/Integer.parseInt("" +ch);
			else
				size--;

		}
		return null;
	}

}
```