CODE EXPLANATION

1. **Self Describing**

Please explain your approach to solving this problem. Would you improve your solution if given more time? If so, how

The algorithm I use was comparing the index and how many times it occurred to the loop of the numbers. the numbers are converted into strings to be able to split.
Position 0 has a value of 2 and there are two 0s in the number
Position 1 has a value of 0 and there are no 1s in the number
Position 2 has value 2 and there are two 2s
Position 3 has a value of 0 and there are zero 3s
The functions return "if" the condition is true or false

What you will do to improve

Instead of looping through the string each time I get the index
I will first loop through the string then I will use a dictionary to keep the value pairs to know how many times the numbers or variables occur e.g 2 occurring twice and zero occurring twice
It would be easier to first convert the integer into a String, then a char array. Loop the array once and use a Map to count. Then loop the second time to compare index and value in the map.

2. **Camel Case**

Please explain your approach to solving this problem. Would you improve your solution if given more time? If so, how

I implemented a count, then stored the answer to know when to start the camel case, then I looped through each input. Based on my count, if count=0 all letters lowercase, if the count is greater than zero first letter upper case and Letter capital. If there's a space my algorithm is to reset count

What you will do to improve

I will skip the substring and try to convert the Letter capital in time complexity 0(n) to make it faster.

Jumbled words explanation

Approach**:** This problem can be solved easily by observing an interesting fact that all even digits have at least one character not present in any other strings, while all odd digits don't:For the odd digits, every letter also appears in some other digit.

- I Created a vector of strings **num** that will hold the numbers in English letter from **0** to **9** and a vector of integers, **count[]** of size **10** and also create an answer string **ans** to display the numbers.
- I, traverse the given string from **i = 0 to N-1**.

If the character is
z’, I increased the zero index by **1**.’
‘X’, I increased the sixth index by **1**.
And so on

- Now, update the elements of the vector as shown below:
- ***count[7] = count[7] – count[6]***.
- ***count[5] = count[5] – count[7]***.
- ***count[4] = count[4] – count[5]***.
- ***count[1] = count[1] – (count[2] + count[4] + count[0])***.
- ***count[3] = count[3] – count[8]***.
- ***count[9] = count[9] – (count[5] + count[6] + count[8])***.
- I, traverse the vector from **0 to 9** and append character **(i + ‘0’)** to **ans**, **count[i]** times.
- Finally, print **ans** as the required answer.