Problem 2 - Triathlon

You are given a text with lower-case letters and you need to perform three different tasks on it – compression, encryption and transformation.

1. **Compression** – you need to reduce the size of the text to the minimum without losing any data. You need to look for sequences of symbols which are the same ('aaaa' for example). For each sequence of same symbols, you need to replace the sequence with a number representing the count of repeated symbols, followed immediately by the symbol, which is repeated. You need to replace symbols, only if the compressed result has less length than the sequence itself. For example you do not need to compress 'a' to '1a' and 'aa' to '2a' because the result is not compressed.
    - Example: '**aaaabbbccccaa**' will be compressed to '**4a3b4caa**'
2. **Encryption** – you need to encrypt the result from the compression with an algorithm. First you need a cypher key. The cypher is easy – each lower-case letter is encrypted with another lower-case. For example 'a' may correspond to 'g', 'b' to 'h', 'c' to 'i', etc. Basically to get the cypher for the whole alphabet you need to shift to the right all letters with an offset, which will be given to you.
    - For example if you have an offset equal to **3**, you will get the following cypher:

Alphabet	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Cypher	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

    - And if you have an offset equal to **8**, you will get the following cypher:

Alphabet	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Cypher	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r

    You need to encrypt all letters one by one and append them to the result by the following formula – you get the ASCII code of the letter and the ASCII code of the corresponding cypher and do a **^ (XOR)** operation on the both numbers. If you have a number to encode, do not do it but instead append it directly to the result. You may use the 'a'.charCodeAt(0) function to get the ASCII code. For example - we have an offset of 3 and the current letter is 'f'. The cypher is 'c'. Corresponding ASCII codes are 102 and 99. 102 ^ 99 = 5. We append 5 to the result.

    - Example: '**4a3b4caa**' with and offset of 3 will become …
        - **4** – we do not encode it, append it to the result -> **4**
        - **a** – cypher is **x** and ASCII codes are 97 and 120. 97 ^ 120 = **25** -> result will become **425**
        - **3** – we do not encode it, append it to the result -> **4253**
        - **b** – cypher is **y** and ASCII codes are 98 and 121. 98 ^ 121 = **27** -> result will become **425327**
        - **4** – we do not encode it, append it to the result -> **4253274**

- **c** – cypher is z and ASCII codes are 99 and 122. **25** -> result will become **425327425**
- **a** – as before result is 25 -> result will become **42532742525**
- **a** – as before result is 25 -> result will become **4253274252525**
- **Encrypted value is '4253274252525'**

3. **Transformation** – you need to transform all the digits from the encrypted result. Find the sum of all even digits and the product of all odd digits in the encryption. This one is easy, right? ☺
    - Example: '**4253274252525**' – even digits are 4, 2, 2, 4, 2, 2, 2. Their sum is **18**. Odd digits are 5, 3, 7, 5, 5, 5. Their product is **13125**.

Log in the console first the **sum** then the **product** on different lines.

Input

The input data is given as a parameter - an array of strings.

The first element in the array will be the original text.

The second element in the array will be the offset of the alphabet cypher.

Output

The output should be printed on the console.

On the first line print the final sum.

On the second line print the final product.

Sample solution code (in JavaScript)

```
function solve(args) {
    var result,
        text = args[0],
        offset = args[1],
        CONSTANTS = {
            ALPHABET: 'abcdefghijklmnopqrstuvwxyz'
        }

    console.log(result);
}
```

Constraints

- The text will contain only lower-case letters
- The text’s length will be no more than **100** symbols
- The offset will be between **0** and **25**
- Allowed working time for your program: **0.2 seconds**
- Allowed memory: **32 MB**

Examples

Input	Output
aaaabbbccccaa 3	18 13125