Problem 2 – Decoding

You are given a number (SALT) and a text. The text must be encoded using the SALT.

The encoding is done as follows:

- If the character is “@”, stop the program
- If the character in the original text is a letter, multiply its char code by the given SALT and add 1000.
- If the character in the original text is a digit, add to its char code SALT and add 500
- If the character in the original text is not a digit or letter (any other symbol), subtract from its char code SALT
- After performing the above operations on the character in the original text:
    - If the character is on even position in the original text - divide the encoded value by 100 and round the precision to 2 decimal digits right of the decimal point
    - Else if the character is on odd position in the original text - multiply by 100
    - The first character is at position 0.

Your task is to encode a text, using the rules above the original text, by given the encoded values for each character.

Input

The input data is given at the standard input.

On the first line you will find an integer number – the SALT number

On the next line, you will find the text, ending with “@”

The input will be valid, in the specified format, within the constraints given below. There is no need to check the input data explicitly.

Output

Print the encoded text

Constraints

- **SALT** will always be between **1 and 10**
- The count of the characters in the text will always be less than **10 000**
- Allowed working time for your program: 0.1 seconds.
- Allowed memory: 16 MB.

Example

Input	Output
9 I love C#!@	16.57 2300 19.72 199900 20.62 190900 0.23 160300

```
0.26
2400
```

Input

```
10
Starwars 4, 5 and 6 are better that
Starwars 1, 2 and 3 :-)@
```

Output

```
18.30
216000
19.70
214000
21.90
197000
21.40
215000
0.22
56200
0.34
2200
5.63
2200
19.70
210000
20.00
2200
5.64
2200
19.70
214000
20.10
2200
19.80
201000
21.60
216000
20.10
214000
0.22
216000
20.40
197000
21.60
2200
18.30
216000
19.70
214000
21.90
197000
21.40
215000
0.22
55900
0.34
2200
5.60
2200
19.70
210000
20.00
2200
5.61
2200
```

	`0.48`
	`3500`
	`0.31`