

Problem A: Factor me in!

Source File: prob_a.cpp | prob_a.java

Input file: prob_a.in

A prime number is an integer greater than 1 whose only factors are 1 and itself. In about 300 BC, Euclid demonstrated that there is an infinite number of prime numbers. What is more amazing is that every natural number greater than or equal to 2 has a unique set of prime factors. Each prime factor of integer n greater than or equal to 2 is a prime number that divides n exactly.

Gregorio read about this discovery of Euclid and tasked himself into writing a program that determines the list of prime factors of a number n greater than or equal to 2.

INPUT FORMAT

The input consists of one or more lines with each line consisting of a positive integer.

OUTPUT FORMAT

For each input line, the output consists of a list of ALL prime factors separated by a single space. The list of prime factors should be ordered in increasing manner. If the input is less than 2, output the message 'not applicable.'

SAMPLE INPUT**SAMPLE OUTPUT**

5	5
1	not applicable
10	2 5
8	2 2 2
231	3 7 11

Problem B: And they are framed up
Source File: prob_b.cpp | prob_b.java
Input file: prob_b.in

Erwin has this problem of enclosing text messages in a frame. To make the problem simple enough, the frame should be composed of asterisks. Help Erwin frame one or more lines of input text messages.

INPUT FORMAT

The input consists of one or more lines of text input.

OUTPUT FORMAT

Each input message should be outputted enclosed in a frame of asterisks.

SAMPLE INPUT

```
hello there
there's reason for every season
*
a
great
```

SAMPLE OUTPUT

```
*****
*hello there*
*****
*****
*there's reason for every season*
*****
***
***
***
***
*a*
***
*****
*great*
*****
```

Problem C: Pssst! Nothing to see here.

Source File: prob_c.cpp | prob_c.java

Input file: prob_c.in

Uncle Julius doesn't want his messages to his generals to be easily intercepted so he devised a not so clever way of encrypting his messages. In a nutshell, each letter in a plain text is replaced by a letter some fixed number of positions down the english alphabet. For example, for a shift of 4, A is replaced by E and B is replaced by F. This substitution cipher wraps around, in the sense that for a shift of 4, W is replaced by A, X is replaced by B, Y is replaced by C, and Z is replaced by D.

Your task is to decode a message given a guessed shift number n and an encrypted sequence of letter.

INPUT FORMAT

The input consists of one or more lines. Each input line should contain a positive number n and an encrypted message. The number n is the guessed shift value.

OUTPUT FORMAT

For each input line, output the "decrypted" message given the guessed shift number n . If a character in the input is not a letter, simply output the character unchanged.

SAMPLE INPUT

```
0 exxegoexsrgi
1 exxegoexsrgi
2 exxegoexsrgi
3 exxegoexsrgi
4 exxegoexsrgi
5 12342
3 wxyzabcd
29 wxyzabcd
```

SAMPLE OUTPUT

```
exxegoexsrgi
dwdfndwrqfh
cvvcemcvqpeg
buubdlbupodf
attackatonce
12342
tuvwxyza
tuvwxyza
```

Problem D: Sum of Evens and Products of Odds

Source File: prob_d.cpp | prob_d.java

Input file: prob_d.in

Given two numbers x and y , determine the sum of even numbers from x to y and the product of odd numbers from x to y .

INPUT FORMAT

The input consists of one or more lines. Each input line consists of two numbers x and y .

OUTPUT FORMAT

For each input line, output the required sum and product separated by a single space. If the first number in the input is greater than the second number then output 'not applicable.'

SAMPLE INPUT

```
3 5
3 3
3 2
1 10
```

SAMPLE OUTPUT

```
4 15
0 3
not applicable
30 945
```