**题目背景**

提高文件的压缩率一直是人们追求的目标。近几年有人提出了这样一种算法，它虽然只是单纯地对文件进行重排，本身并不压缩文件，但是经这种算法调整后的文件在大多数情况下都能获得比原来更大的压缩率。

**题目描述**

该算法具体如下：对一个长度为 *n* 的字符串 *S*，首先根据它构造 *n* 个字符串，其中第 *i* 个字符串由将 *S* 的前 *i*−1 个字符置于末尾得到。然后把这 *n* 个字符串按照首字符从小到大排序，如果两个字符串的首字符相等，则按照它们在 *S* 中的位置从小到大排序。排序后的字符串的尾字符可以组成一个新的字符串 *S*′ ，它的长度也是 *n*，并且包含了 *S* 中的每一个字符。最后输出 *S*′ 以及 *S* 的首字符在 *S*′ 中的位置 *p*。

举例：*S* 是 example

1. 构造 *n* 个字符串。

example

xamplee

ampleex

mpleexa

pleexam

leexamp

eexampl

1. 将字符串排序。

ampleex

example

eexampl

leexamp

mpleexa

pleexam

xamplee

1. 压缩结果。

S' = xelpame

p = 7

由于英语单词构造的特殊性，某些字母对出现的频率很高，因此在 *S*′ 中相同的字母有很大几率排在一起，从而提高 *S*′ 的压缩率。虽然这种算法利用了英语单词的特性，然而在实践的过程中，人们发现它几乎适用于所有的文件压缩。

请你编一个程序，读入 *S*′ 和 *p*，输出字符串 *S*。

**输入格式**

共三行。

第一行是一个整数 n(1\len\le10000)，代表 *S*′ 的长度。

第二行是字符串 *S*′。

第三行是整数 *p*。

**输出格式**

一行，*S*。

**输入输出样例**

**输入 #1**复制

7

xelpame

7

**输出 #1**复制

example