Problem Statement (STable)

You are given two Strings **s** and **t**. All characters of **s** and **t** are distinct. No character of **s** is present in **t** and no character of **t** is present in **s**.

Let N be the length of **s**, and M be the length of **t**. Define a 2-dimensional string array "table" as follows:

- table[i][0] = **s**[i-1] (1<=i<=N)
- table[0][j] = **t**[j-1] (1<=j<=M)
- table[i][j] = min(table[i-1][j], table[i][j-1]) + max(table[i-1][j], table[i][j-1]) (1<=i<=N, 1<=j<=M)

Note that min and max are defined by the lexicographical order of strings (see Notes for a more formal definition), and A+B means the concatenation of strings A and B.

Your task is to find a substring of table[N][M]. Let L be the length of table[N][M]. Return the substring of table[N][M] whose start position (0-indexed) is **pos** and length is min(50, L-**pos**).

Definition

Class:	STable
Method:	getString
Parameters:	String, String, long
Returns:	String
Method signature:	String getString(String s, String t, long pos)

(be sure your method is public)

Notes

- A string X is defined as smaller than a string Y if and only if X is a prefix of Y or X has a smaller character than Y at the first position where they differ.
- The order of characters is defined by their ASCII codes: '0'<...<'9'<'A'<...<'Z'<'a'<...<'z'.

Constraints

- **s** and **t** will each contain between 1 and 30 characters, inclusive.
- All characters of **s** and **t** will be distinct.-No character of **s** will be present in **t**.
- No character of **t** will be present in **s**.-**s** and **t** will contain only letters ('A'-'Z', 'a'-'z') and digits ('0'-'9').
- **pos** will be between 0 and L-1, inclusive, where L is the length of table[N][M] as defined in the statement.

Examples

0)

```
"ad"

"cb"

0
```

```
Returns: "acbacd"
```

In this case, the array "table" is as follows.

```
-----------------
| |  c   |  b   |
-----------------
|a|  ac  | acb  |
-----------------
|d| acd  |acbacd|
-----------------
```

1)

```
"fox"

"cat"

0
```

```
Returns: "acfcfoacftacfcfocfox"
```

2)

```
"Ra6b1t"

"W0lf"

66
```

```
Returns: "RWab0RWRWa0RWl0RWRWa6RWa0RWRWa6RWa6RWab0RWRWa6RWa6"
```

In this case, return 50 characters.