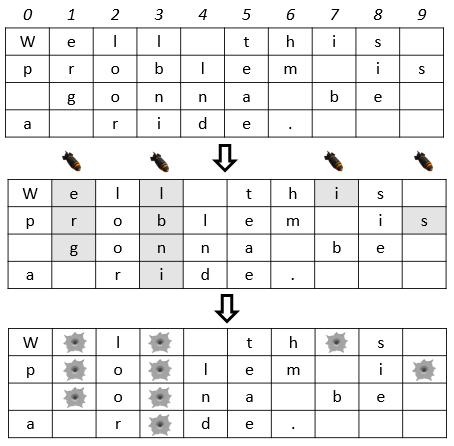
**Problem 4 – Text Bombardment**

Write a program that reads a **text** and **line width** from the console. The program should distribute the text so that it fits in a table with a specific line width. Each cell should contain only **1 character**. It should then read a **line with numbers**, holding the **columns that should be bombarded**.

For example, we read the text "**Well this problem is gonna be a ride.**" and line width **10**. We distribute the text among 4 rows with 10 columns. We read the numbers "**1 3 7 9**" and drop bombs on those columns in the table.

The bombs **destroy** the character they fall on + all the neighboring characters **below** it. **Note**: Empty spaces below destroyed characters stop the bombs (see column 7).

Finally, we print the bombarded text on the console: "**W l th s p o lem i o na be a r de.**"

**Note:** The empty cells in the table after the text should NOT be printed.

**Input**

The input data is read from the console.

* On the first line you will be given the **text**
* On the next lines you will be given the **line width**
* On the third line you will receive the **columns** that should be bombed (space-separated)

The input data will always be valid and in the format described. There is no need to check it explicitly.

**Output**

The output data must be printed on the console and should contain only 1 line: the **bombarded text** as a single string.

**Constraints**

* The text will contain only ASCII characters and will be no longer than 1000 symbols.
* The line width will be in the range [1…100].
* The columns will be valid integers in the range [1…<line width> - 1].
* A column will not be bombed more than once.
* Time limit: 0.25 seconds. Allowed memory: 16 MB.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| Well this problem is gonna be a ride.  10  1 3 7 9 | W l th s p o lem i o na be a r de. |