# Question

You are given a license key represented as a string S which consists only alphanumeric character and dashes. The string is separated into N+1 groups by N dashes.

Given a number K, we would want to reformat the strings such that each group contains *exactly* K characters, except for the first group which could be shorter than K, but still must contain at least one character. Furthermore, there must be a dash inserted between two groups and all lowercase letters should be converted to uppercase.

Given a non-empty string S and a number K, format the string according to the rules described above.

**Example 1:**

**Input:** S = "5F3Z-2e-9-w", K = 4

**Output:** "5F3Z-2E9W"

**Explanation:** The string S has been split into two parts, each part has 4 characters.

Note that the two extra dashes are not needed and can be removed.

**Example 2:**

**Input:** S = "2-5g-3-J", K = 2

**Output:** "2-5G-3J"

**Explanation:** The string S has been split into three parts, each part has 2 characters except the first part as it could be shorter as mentioned above.

**Note:**

1. The length of string S will not exceed 12,000, and K is a positive integer.
2. String S consists only of alphanumerical characters (a-z and/or A-Z and/or 0-9) and dashes(-).
3. String S is non-empty.

# Solution

Java

|  |
| --- |
| public String licenseKeyFormatting(String s, int k) {  StringBuilder sb = new StringBuilder();  for (int i = s.length() - 1; i >= 0; i--)  if (s.charAt(i) != '-')  sb.append(sb.length() % (k + 1) == k ? '-' : "").append(s.charAt(i));  return sb.reverse().toString().toUpperCase();  } |

Python

|  |
| --- |
| class Solution:  def licenseKeyFormatting(self, S, K):  """  :type S: str  :type K: int  :rtype: str  """  S = S.replace("-", "").upper()[::-1]  return '-'.join(S[i:i+K] for i in range(0, len(S), K))[::-1] |