Problem 482: License Key Formatting

Problem Information

Difficulty: Easy
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

You are given a license key represented as a string

s

that consists of only alphanumeric characters and dashes. The string is separated into

n + 1

groups by

n

dashes. You are also given an integer

k

.

We want to reformat the string

s

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 you should convert all lowercase letters to uppercase.

Return

the reformatted license key

.

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.

Constraints:

1 <= s.length <= 10

5

s

consists of English letters, digits, and dashes

'-'

.

1 <= k <= 10

4

Code Snippets

C++:

```
class Solution {
public:
string licenseKeyFormatting(string s, int k) {

}
};
```

Java:

```
class Solution {
public String licenseKeyFormatting(String s, int k) {

}
}
```

Python3:

```
class Solution:
def licenseKeyFormatting(self, s: str, k: int) -> str:
```

Python:

```
class Solution(object):
def licenseKeyFormatting(self, s, k):
"""
:type s: str
:type k: int
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} s
* @param {number} k
* @return {string}
*/
var licenseKeyFormatting = function(s, k) {

};
```

TypeScript:

```
function licenseKeyFormatting(s: string, k: number): string {

};
```

C#:

```
public class Solution {
public string LicenseKeyFormatting(string s, int k) {

}
}
```

C:

```
char* licenseKeyFormatting(char* s, int k) {

}
```

Go:

```
func licenseKeyFormatting(s string, k int) string {

}
```

Kotlin:

```
class Solution {
fun licenseKeyFormatting(s: String, k: Int): String {

}
}
```

Swift:

```
class Solution {
func licenseKeyFormatting(_ s: String, _ k: Int) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn license_key_formatting(s: String, k: i32) -> String {

}
}
```

Ruby:

```
# @param {String} s
# @param {Integer} k
# @return {String}
def license_key_formatting(s, k)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @param Integer $k
* @return String
*/
function licenseKeyFormatting($s, $k) {

}
}
```

Dart:

```
class Solution {
String licenseKeyFormatting(String s, int k) {

}
}
```

Scala:

```
object Solution {
def licenseKeyFormatting(s: String, k: Int): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec license_key_formatting(s :: String.t, k :: integer) :: String.t
def license_key_formatting(s, k) do

end
end
```

Erlang:

```
-spec license_key_formatting(S :: unicode:unicode_binary(), K :: integer())
-> unicode:unicode_binary().
```

```
license_key_formatting(S, K) ->
.
```

Racket:

```
(define/contract (license-key-formatting s k)
(-> string? exact-integer? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: License Key Formatting
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
string licenseKeyFormatting(string s, int k) {

}
};
```

Java Solution:

```
/**
* Problem: License Key Formatting
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
```

```
*/

class Solution {
public String licenseKeyFormatting(String s, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: License Key Formatting
Difficulty: Easy
Tags: string

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def licenseKeyFormatting(self, s: str, k: int) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def licenseKeyFormatting(self, s, k):
"""
:type s: str
:type k: int
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: License Key Formatting
* Difficulty: Easy
* Tags: string
```

```
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {string} s
* @param {number} k
* @return {string}
*/
var licenseKeyFormatting = function(s, k) {

};
```

TypeScript Solution:

```
/**
* Problem: License Key Formatting
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function licenseKeyFormatting(s: string, k: number): string {

};
```

C# Solution:

```
/*
* Problem: License Key Formatting
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
```

```
*/

public class Solution {
public string LicenseKeyFormatting(string s, int k) {

}
}
```

C Solution:

```
/*
* Problem: License Key Formatting
* Difficulty: Easy
* Tags: string
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

char* licenseKeyFormatting(char* s, int k) {

}
```

Go Solution:

```
// Problem: License Key Formatting
// Difficulty: Easy
// Tags: string
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func licenseKeyFormatting(s string, k int) string {

}
```

Kotlin Solution:

```
class Solution {
fun licenseKeyFormatting(s: String, k: Int): String {

}
}
```

Swift Solution:

```
class Solution {
func licenseKeyFormatting(_ s: String, _ k: Int) -> String {

}
}
```

Rust Solution:

```
// Problem: License Key Formatting
// Difficulty: Easy
// Tags: string
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn license_key_formatting(s: String, k: i32) -> String {

}
}
```

Ruby Solution:

```
# @param {String} s
# @param {Integer} k
# @return {String}
def license_key_formatting(s, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @param Integer $k
* @return String
*/
function licenseKeyFormatting($s, $k) {

}
}
```

Dart Solution:

```
class Solution {
String licenseKeyFormatting(String s, int k) {

}
}
```

Scala Solution:

```
object Solution {
def licenseKeyFormatting(s: String, k: Int): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec license_key_formatting(s :: String.t, k :: integer) :: String.t
def license_key_formatting(s, k) do

end
end
```

Erlang Solution:

```
-spec license_key_formatting(S :: unicode:unicode_binary(), K :: integer())
-> unicode:unicode_binary().
license_key_formatting(S, K) ->
```

```
.
```

Racket Solution:

```
(define/contract (license-key-formatting s k)
(-> string? exact-integer? string?)
)
```