Problem 1694: Reformat Phone Number

Problem Information

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

Problem Description

You are given a phone number as a string

number

.

number

consists of digits, spaces

' '

, and/or dashes

'-'

.

You would like to reformat the phone number in a certain manner. Firstly,

remove

all spaces and dashes. Then,

group

the digits from left to right into blocks of length 3

until

there are 4 or fewer digits. The final digits are then grouped as follows:

2 digits: A single block of length 2.

3 digits: A single block of length 3.

4 digits: Two blocks of length 2 each.

The blocks are then joined by dashes. Notice that the reformatting process should

never

produce any blocks of length 1 and produce

at most

two blocks of length 2.

Return

the phone number after formatting.

Example 1:

Input:

number = "1-23-45 6"

Output:

"123-456"

Explanation:

The digits are "123456". Step 1: There are more than 4 digits, so group the next 3 digits. The 1st block is "123". Step 2: There are 3 digits remaining, so put them in a single block of length 3. The 2nd block is "456". Joining the blocks gives "123-456".

Example 2:

Input:

number = "123 4-567"

Output:

"123-45-67"

Explanation:

The digits are "1234567". Step 1: There are more than 4 digits, so group the next 3 digits. The 1st block is "123". Step 2: There are 4 digits left, so split them into two blocks of length 2. The blocks are "45" and "67". Joining the blocks gives "123-45-67".

Example 3:

Input:

number = "123 4-5678"

Output:

"123-456-78"

Explanation:

The digits are "12345678". Step 1: The 1st block is "123". Step 2: The 2nd block is "456". Step 3: There are 2 digits left, so put them in a single block of length 2. The 3rd block is "78". Joining the blocks gives "123-456-78".

Constraints:

2 <= number.length <= 100

number

consists of digits and the characters

'-'

and

' '

.

There are at least

two

digits in

number

.

Code Snippets

C++:

```
class Solution {
public:
string reformatNumber(string number) {

}
};
```

Java:

```
class Solution {
public String reformatNumber(String number) {

}
}
```

Python3:

```
class Solution:
def reformatNumber(self, number: str) -> str:
```

Python:

```
class Solution(object):
def reformatNumber(self, number):
"""
:type number: str
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} number
* @return {string}
*/
var reformatNumber = function(number) {

};
```

TypeScript:

```
function reformatNumber(number: string): string {

};
```

C#:

```
public class Solution {
public string ReformatNumber(string number) {

}
}
```

C:

```
char* reformatNumber(char* number) {

}
```

Go:

```
func reformatNumber(number string) string {

}
```

Kotlin:

```
class Solution {
fun reformatNumber(number: String): String {

}
}
```

Swift:

```
class Solution {
func reformatNumber(_ number: String) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn reformat_number(number: String) -> String {

}
}
```

Ruby:

```
# @param {String} number
# @return {String}
def reformat_number(number)

end
```

PHP:

```
class Solution {

/**
* @param String $number
* @return String
*/
function reformatNumber($number) {

}
}
```

Dart:

```
class Solution {
String reformatNumber(String number) {

}
}
```

Scala:

```
object Solution {
def reformatNumber(number: String): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec reformat_number(number :: String.t) :: String.t
def reformat_number(number) do

end
end
```

Erlang:

```
-spec reformat_number(Number :: unicode:unicode_binary()) ->
unicode:unicode_binary().
reformat_number(Number) ->
.
```

Racket:

```
(define/contract (reformat-number number)
(-> string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Reformat Phone Number
* 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 reformatNumber(string number) {

}
};
```

Java Solution:

```
/**
* Problem: Reformat Phone Number
* 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 reformatNumber(String number) {
}
}
```

Python3 Solution:

```
"""
Problem: Reformat Phone Number
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 reformatNumber(self, number: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def reformatNumber(self, number):
"""
:type number: str
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Reformat Phone Number
* 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} number
* @return {string}
*/
var reformatNumber = function(number) {

};
```

TypeScript Solution:

```
/**
* Problem: Reformat Phone Number
* 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 reformatNumber(number: string): string {

};
```

C# Solution:

```
/*
* Problem: Reformat Phone Number
* 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 ReformatNumber(string number) {

}
}
```

C Solution:

```
/*
* Problem: Reformat Phone Number
* 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* reformatNumber(char* number) {

}
```

Go Solution:

```
// Problem: Reformat Phone Number
// 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 reformatNumber(number string) string {

}
```

Kotlin Solution:

```
class Solution {
fun reformatNumber(number: String): String {

}
}
```

Swift Solution:

```
class Solution {
func reformatNumber(_ number: String) -> String {

}
}
```

Rust Solution:

```
// Problem: Reformat Phone Number
// 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 reformat_number(number: String) -> String {

}
}
```

Ruby Solution:

```
# @param {String} number
# @return {String}
def reformat_number(number)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $number
* @return String
*/
function reformatNumber($number) {

}
}
```

Dart Solution:

```
class Solution {
String reformatNumber(String number) {

}
}
```

Scala Solution:

```
object Solution {
def reformatNumber(number: String): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec reformat_number(number :: String.t) :: String.t
def reformat_number(number) do

end
end
```

Erlang Solution:

```
-spec reformat_number(Number :: unicode:unicode_binary()) ->
unicode:unicode_binary().
reformat_number(Number) ->
.
```

Racket Solution:

```
(define/contract (reformat-number number)
(-> string? string?)
)
```