Problem 831: Masking Personal Information

Problem Information

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

Problem Description

You are given a personal information string

s

, representing either an

email address

or a

phone number

. Return

the

masked

personal information using the below rules

.

Email address:

An email address is:

A

name

consisting of uppercase and lowercase English letters, followed by

The

'@'

symbol, followed by

The

domain

consisting of uppercase and lowercase English letters with a dot

'.'

somewhere in the middle (not the first or last character).

To mask an email:

The uppercase letters in the

name

and

domain

must be converted to lowercase letters.

The middle letters of the

name

(i.e., all but the first and last letters) must be replaced by 5 asterisks

"*****"

.

Phone number:

A phone number is formatted as follows:

The phone number contains 10-13 digits.

The last 10 digits make up the

local number

.

The remaining 0-3 digits, in the beginning, make up the

country code

.

Separation characters

from the set

{'+', '-', '(', ')', ' '}

separate the above digits in some way.

To mask a phone number:

Remove all

separation characters

.

The masked phone number should have the form:

"***-***-XXXX"

if the country code has 0 digits.

"+*-***-***-XXXX"

if the country code has 1 digit.

"+**-***-***-XXXX"

if the country code has 2 digits.

"+***-***-***-XXXX"

if the country code has 3 digits.

"XXXX"

is the last 4 digits of the

local number

.

Example 1:

Input:

s = "LeetCode@LeetCode.com"

Output:

"l*****e@leetcode.com"

Explanation:

s is an email address. The name and domain are converted to lowercase, and the middle of the name is replaced by 5 asterisks.

Example 2:

Input:

s = "AB@qq.com"

Output:

"a*****b@qq.com"

Explanation:

s is an email address. The name and domain are converted to lowercase, and the middle of the name is replaced by 5 asterisks. Note that even though "ab" is 2 characters, it still must have 5 asterisks in the middle.

Example 3:

Input:

s = "1(234)567-890"

Output:

"***-***-7890"

Explanation:

s is a phone number. There are 10 digits, so the local number is 10 digits and the country code is 0 digits. Thus, the resulting masked number is "***-***-7890".

Constraints:

s

is either a

valid

email or a phone number.

If

s

is an email:

8 <= s.length <= 40

s

consists of uppercase and lowercase English letters and exactly one

'@'

symbol and

'.'

symbol.

If

s

is a phone number:

10 <= s.length <= 20

s

consists of digits, spaces, and the symbols

'('

,

')'

,

'-'

, and

'+'

.

Code Snippets

C++:

```
class Solution {
public:
string maskPII(string s) {

}
};
```

Java:

```
class Solution {
public String maskPII(String s) {

}
}
```

Python3:

```
class Solution:
def maskPII(self, s: str) -> str:
```

Python:

```
class Solution(object):
def maskPII(self, s):
"""
:type s: str
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} s
* @return {string}
*/
var maskPII = function(s) {

};
```

TypeScript:

```
function maskPII(s: string): string {

};
```

C#:

```
public class Solution {
public string MaskPII(string s) {

}
}
```

C:

```
char* maskPII(char* s) {

}
```

Go:

```
func maskPII(s string) string {

}
```

Kotlin:

```
class Solution {
fun maskPII(s: String): String {

}
}
```

Swift:

```
class Solution {
func maskPII(_ s: String) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn mask_pii(s: String) -> String {

}
}
```

Ruby:

```
# @param {String} s
# @return {String}
def mask_pii(s)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @return String
*/
function maskPII($s) {

}
```

```
}
```

Dart:

```
class Solution {
String maskPII(String s) {

}
}
```

Scala:

```
object Solution {
def maskPII(s: String): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec mask_pii(s :: String.t) :: String.t
def mask_pii(s) do

end
end
```

Erlang:

```
-spec mask_pii(S :: unicode:unicode_binary()) -> unicode:unicode_binary().
mask_pii(S) ->
.
```

Racket:

```
(define/contract (mask-pii s)
(-> string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Masking Personal Information
* Difficulty: Medium
* 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 maskPII(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Masking Personal Information
* Difficulty: Medium
* 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 maskPII(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Masking Personal Information
Difficulty: Medium
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 maskPII(self, s: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def maskPII(self, s):
"""
:type s: str
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Masking Personal Information
* Difficulty: Medium
* 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
* @return {string}
*/
var maskPII = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Masking Personal Information
* Difficulty: Medium
* 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 maskPII(s: string): string {

};
```

C# Solution:

```
/*
* Problem: Masking Personal Information
* Difficulty: Medium
* 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 MaskPII(string s) {

}
}
```

C Solution:

```
/*
* Problem: Masking Personal Information
* Difficulty: Medium
* 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* maskPII(char* s) {

}
```

Go Solution:

```
// Problem: Masking Personal Information
// Difficulty: Medium
// 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 maskPII(s string) string {

}
```

Kotlin Solution:

```
class Solution {
fun maskPII(s: String): String {

}
}
```

Swift Solution:

```
class Solution {
func maskPII(_ s: String) -> String {

}
}
```

Rust Solution:

```
// Problem: Masking Personal Information
// Difficulty: Medium
// 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 mask_pii(s: String) -> String {

}
}
```

Ruby Solution:

```
# @param {String} s
# @return {String}
def mask_pii(s)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @return String
*/
function maskPII($s) {

}
}
```

Dart Solution:

```
class Solution {
String maskPII(String s) {

}
}
```

Scala Solution:

```
object Solution {
def maskPII(s: String): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec mask_pii(s :: String.t) :: String.t
def mask_pii(s) do

end
end
```

Erlang Solution:

```
-spec mask_pii(S :: unicode:unicode_binary()) -> unicode:unicode_binary().
mask_pii(S) ->
.
```

Racket Solution:

```
(define/contract (mask-pii s)
(-> string? string?)
)
```