Problem 3174: Clear Digits

Problem Information

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

Problem Description

You are given a string

s

.

Your task is to remove

all

digits by doing this operation repeatedly:

Delete the

first

digit and the

closest

non-digit

character to its

left

.

Return the resulting string after removing all digits.

Note

that the operation

cannot

be performed on a digit that does not have any non-digit character to its left.

Example 1:

Input:

s = "abc"

Output:

"abc"

Explanation:

There is no digit in the string.

Example 2:

Input:

s = "cb34"

Output:

""

Explanation:

First, we apply the operation on

s[2]

, and

s

becomes

"c4"

.

Then we apply the operation on

s[1]

, and

s

becomes

""

.

Constraints:

1 <= s.length <= 100

s

consists only of lowercase English letters and digits.

The input is generated such that it is possible to delete all digits.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
char* clearDigits(char* s) {

}
```

Go:

```
func clearDigits(s string) string {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (clear-digits s)
(-> string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Clear Digits
* Difficulty: Easy
* Tags: string, stack
*
* 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 clearDigits(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Clear Digits
* Difficulty: Easy
* Tags: string, stack
*
* 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 clearDigits(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Clear Digits
Difficulty: Easy
Tags: string, stack

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 clearDigits(self, s: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

```
"""
```

JavaScript Solution:

```
/**
* Problem: Clear Digits
* Difficulty: Easy
* Tags: string, stack
*
* 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 clearDigits = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Clear Digits
* Difficulty: Easy
* Tags: string, stack
*
* 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 clearDigits(s: string): string {

};
```

C# Solution:

```
/*
* Problem: Clear Digits
* Difficulty: Easy
* Tags: string, stack
*
* 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 ClearDigits(string s) {

}
}
```

C Solution:

```
/*
* Problem: Clear Digits
* Difficulty: Easy
* Tags: string, stack
*
* 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* clearDigits(char* s) {

}
```

Go Solution:

```
// Problem: Clear Digits
// Difficulty: Easy
// Tags: string, stack
//
// 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 clearDigits(s string) string {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Clear Digits
// Difficulty: Easy
// Tags: string, stack
//
// 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 clear_digits(s: String) -> String {

}
}
```

Ruby Solution:

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

```
end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (clear-digits s)
(-> string? string?)
)
```