Problem 1844: Replace All Digits with Characters

Problem Information

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

Problem Description

You are given a

0-indexed

string

s

that has lowercase English letters in its

even

indices and digits in its

odd

indices.

You must perform an operation

shift(c, x)

, where

c

is a character and

x

is a digit, that returns the

x

th

character after

c

.

For example,

shift('a', 5) = 'f'

and

shift('x', 0) = 'x'

.

For every

odd

index

i

, you want to replace the digit

s[i]

with the result of the

shift(s[i-1], s[i])

operation.

Return

s

after replacing all digits. It is

guaranteed

that

shift(s[i-1], s[i])

will never exceed

'z'

.

Note

that

shift(c, x)

is

not

a preloaded function, but an operation

to be implemented

as part of the solution.

Example 1:

Input:

s = "a1c1e1"

Output:

"abcdef"

Explanation:

The digits are replaced as follows: - s[1] -> shift('a',1) = 'b' - s[3] -> shift('c',1) = 'd' - s[5] -> shift('e',1) = 'f'

Example 2:

Input:

s = "a1b2c3d4e"

Output:

"abbdcfdhe"

Explanation:

The digits are replaced as follows: - s[1] -> shift('a',1) = 'b' - s[3] -> shift('b',2) = 'd' - s[5] -> shift('c',3) = 'f' - s[7] -> shift('d',4) = 'h'

Constraints:

1 <= s.length <= 100

s

consists only of lowercase English letters and digits.

shift(s[i-1], s[i]) <= 'z'

for all

odd

indices

i

.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

```
:rtype: str
"""
```

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
char* replaceDigits(char* s) {

}
```

Go:

```
func replaceDigits(s string) string {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Replace All Digits with Characters
* 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 replaceDigits(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Replace All Digits with Characters
* 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 replaceDigits(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Replace All Digits with Characters
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 replaceDigits(self, s: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Replace All Digits with Characters
* 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
* @return {string}
*/
var replaceDigits = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Replace All Digits with Characters
* 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 replaceDigits(s: string): string {

};
```

C# Solution:

```
/*
* Problem: Replace All Digits with Characters
* 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 ReplaceDigits(string s) {

}
}
```

C Solution:

```
/*
* Problem: Replace All Digits with Characters
* 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* replaceDigits(char* s) {

}
```

Go Solution:

```
// Problem: Replace All Digits with Characters
// 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 replaceDigits(s string) string {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Replace All Digits with Characters
// 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 replace_digits(s: String) -> String {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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