Problem 1487: Making File Names Unique

Problem Information

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

Problem Description

Given an array of strings

names

of size

n

. You will create

n

folders in your file system

such that

, at the

i

th

minute, you will create a folder with the name

names[i]

.

Since two files

cannot

have the same name, if you enter a folder name that was previously used, the system will have a suffix addition to its name in the form of

(k)

, where,

k

is the

smallest positive integer

such that the obtained name remains unique.

Return

an array of strings of length

n

where

ans[i]

is the actual name the system will assign to the

i

th

folder when you create it.

Example 1:

Input:

names = ["pes","fifa","gta","pes(2019)"]

Output:

["pes","fifa","gta","pes(2019)"]

Explanation:

Let's see how the file system creates folder names: "pes" --> not assigned before, remains "pes" "fifa" --> not assigned before, remains "fifa" "gta" --> not assigned before, remains "gta" "pes(2019)" --> not assigned before, remains "pes(2019)"

Example 2:

Input:

names = ["gta","gta(1)","gta","avalon"]

Output:

["gta","gta(1)","gta(2)","avalon"]

Explanation:

Let's see how the file system creates folder names: "gta" --> not assigned before, remains "gta" "gta(1)" --> not assigned before, remains "gta(1)" "gta" --> the name is reserved, system adds (k), since "gta(1)" is also reserved, systems put k = 2. it becomes "gta(2)" "avalon" --> not assigned before, remains "avalon"

Example 3:

Input:

names = ["onepiece","onepiece(1)","onepiece(2)","onepiece(3)","onepiece"]

Output:

["onepiece","onepiece(1)","onepiece(2)","onepiece(3)","onepiece(4)"]

Explanation:

When the last folder is created, the smallest positive valid k is 4, and it becomes "onepiece(4)".

Constraints:

1 <= names.length <= 5 * 10

4

1 <= names[i].length <= 20

names[i]

consists of lowercase English letters, digits, and/or round brackets.

Code Snippets

C++:

```
class Solution {
public:
vector<string> getFolderNames(vector<string>& names) {

}
};
```

Java:

```
class Solution {
public String[] getFolderNames(String[] names) {

}
}
```

Python3:

```
class Solution:
def getFolderNames(self, names: List[str]) -> List[str]:
```

Python:

```
class Solution(object):
def getFolderNames(self, names):
"""
:type names: List[str]
:rtype: List[str]
"""
```

JavaScript:

```
/**
* @param {string[]} names
* @return {string[]}
*/
var getFolderNames = function(names) {

};
```

TypeScript:

```
function getFolderNames(names: string[]): string[] {

};
```

C#:

```
public class Solution {
public string[] GetFolderNames(string[] names) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
```

```
char** getFolderNames(char** names, int namesSize, int* returnSize) {

}
```

Go:

```
func getFolderNames(names []string) []string {

}
```

Kotlin:

```
class Solution {
fun getFolderNames(names: Array<String>): Array<String> {

}
}
```

Swift:

```
class Solution {
func getFolderNames(_ names: [String]) -> [String] {

}
}
```

Rust:

```
impl Solution {
pub fn get_folder_names(names: Vec<String>) -> Vec<String> {

}
}
```

Ruby:

```
# @param {String[]} names
# @return {String[]}
def get_folder_names(names)

end
```

PHP:

```
class Solution {

/**
* @param String[] $names
* @return String[]
*/
function getFolderNames($names) {

}
}
```

Dart:

```
class Solution {
List<String> getFolderNames(List<String> names) {

}
}
```

Scala:

```
object Solution {
def getFolderNames(names: Array[String]): Array[String] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec get_folder_names(names :: [String.t]) :: [String.t]
def get_folder_names(names) do

end
end
```

Erlang:

```
-spec get_folder_names(Names :: [unicode:unicode_binary()]) ->
[unicode:unicode_binary()].
get_folder_names(Names) ->
```

```
.
```

Racket:

```
(define/contract (get-folder-names names)
(-> (listof string?) (listof string?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Making File Names Unique
* Difficulty: Medium
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
vector<string> getFolderNames(vector<string>& names) {

}
};
```

Java Solution:

```
/**
* Problem: Making File Names Unique
* Difficulty: Medium
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/
```

```
class Solution {
public String[] getFolderNames(String[] names) {

}
}
```

Python3 Solution:

```
"""
Problem: Making File Names Unique
Difficulty: Medium
Tags: array, string, hash

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class Solution:
def getFolderNames(self, names: List[str]) -> List[str]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def getFolderNames(self, names):
"""
:type names: List[str]
:rtype: List[str]
"""
```

JavaScript Solution:

```
/**
* Problem: Making File Names Unique
* Difficulty: Medium
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
```

```
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {string[]} names
* @return {string[]}
*/
var getFolderNames = function(names) {

};
```

TypeScript Solution:

```
/**
* Problem: Making File Names Unique
* Difficulty: Medium
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function getFolderNames(names: string[]): string[] {

};
```

C# Solution:

```
/*
* Problem: Making File Names Unique
* Difficulty: Medium
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
```

```
public string[] GetFolderNames(string[] names) {

}
}
```

C Solution:

```
/*
* Problem: Making File Names Unique
* Difficulty: Medium
* Tags: array, string, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* Note: The returned array must be malloced, assume caller calls free().
*/
char** getFolderNames(char** names, int namesSize, int* returnSize) {

}
```

Go Solution:

```
// Problem: Making File Names Unique
// Difficulty: Medium
// Tags: array, string, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func getFolderNames(names []string) []string {

}
```

Kotlin Solution:

```
class Solution {
fun getFolderNames(names: Array<String>): Array<String> {

}
}
```

Swift Solution:

```
class Solution {
func getFolderNames(_ names: [String]) -> [String] {

}
}
```

Rust Solution:

```
// Problem: Making File Names Unique
// Difficulty: Medium
// Tags: array, string, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn get_folder_names(names: Vec<String>) -> Vec<String> {

}
}
```

Ruby Solution:

```
# @param {String[]} names
# @return {String[]}
def get_folder_names(names)

end
```

PHP Solution:

```
class Solution {
```

```
/**
* @param String[] $names
* @return String[]
*/
function getFolderNames($names) {

}
}
```

Dart Solution:

```
class Solution {
List<String> getFolderNames(List<String> names) {

}
}
```

Scala Solution:

```
object Solution {
def getFolderNames(names: Array[String]): Array[String] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec get_folder_names(names :: [String.t]) :: [String.t]
def get_folder_names(names) do

end
end
```

Erlang Solution:

```
-spec get_folder_names(Names :: [unicode:unicode_binary()]) ->
[unicode:unicode_binary()].
get_folder_names(Names) ->
.
```

Racket Solution:

```
(define/contract (get-folder-names names)
(-> (listof string?) (listof string?))
)
```