Problem 925: Long Pressed Name

Problem Information

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

Problem Description

Your friend is typing his

name

into a keyboard. Sometimes, when typing a character

c

, the key might get

long pressed

, and the character will be typed 1 or more times.

You examine the

typed

characters of the keyboard. Return

True

if it is possible that it was your friends name, with some characters (possibly none) being long pressed.

Example 1:

Input:

name = "alex", typed = "aaleex"

Output:

true

Explanation:

'a' and 'e' in 'alex' were long pressed.

Example 2:

Input:

name = "saeed", typed = "ssaaedd"

Output:

false

Explanation:

'e' must have been pressed twice, but it was not in the typed output.

Constraints:

1 <= name.length, typed.length <= 1000

name

and

typed

consist of only lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
bool isLongPressedName(string name, string typed) {

}
};
```

Java:

```
class Solution {
public boolean isLongPressedName(String name, String typed) {

}
}
```

Python3:

```
class Solution:
def isLongPressedName(self, name: str, typed: str) -> bool:
```

Python:

```
class Solution(object):
def isLongPressedName(self, name, typed):
"""
:type name: str
:type typed: str
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {string} name
* @param {string} typed
* @return {boolean}
*/
var isLongPressedName = function(name, typed) {
```

```
};
```

TypeScript:

```
function isLongPressedName(name: string, typed: string): boolean {

};
```

C#:

```
public class Solution {
public bool IsLongPressedName(string name, string typed) {

}
}
```

C:

```
bool isLongPressedName(char* name, char* typed) {

}
```

Go:

```
func isLongPressedName(name string, typed string) bool {

}
```

Kotlin:

```
class Solution {
fun isLongPressedName(name: String, typed: String): Boolean {

}
}
```

Swift:

```
class Solution {
func isLongPressedName(_ name: String, _ typed: String) -> Bool {
```

```
}
}
```

Rust:

```
impl Solution {
pub fn is_long_pressed_name(name: String, typed: String) -> bool {

}
}
```

Ruby:

```
# @param {String} name
# @param {String} typed
# @return {Boolean}
def is_long_pressed_name(name, typed)

end
```

PHP:

```
class Solution {

/**
* @param String $name
* @param String $typed
* @return Boolean
*/
function isLongPressedName($name, $typed) {

}
}
```

Dart:

```
class Solution {
bool isLongPressedName(String name, String typed) {

}
}
```

Scala:

```
object Solution {
def isLongPressedName(name: String, typed: String): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec is_long_pressed_name(name :: String.t, typed :: String.t) :: boolean
def is_long_pressed_name(name, typed) do

end
end
```

Erlang:

```
-spec is_long_pressed_name(Name :: unicode:unicode_binary(), Typed ::
unicode:unicode_binary()) -> boolean().
is_long_pressed_name(Name, Typed) ->
.
```

Racket:

```
(define/contract (is-long-pressed-name name typed)
(-> string? string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Long Pressed Name
* Difficulty: Easy
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
```

```
*/

class Solution {
public:
bool isLongPressedName(string name, string typed) {

}
};
```

Java Solution:

```
/**
* Problem: Long Pressed Name
* Difficulty: Easy
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public boolean isLongPressedName(String name, String typed) {

}
}
```

Python3 Solution:

```
"""
Problem: Long Pressed Name
Difficulty: Easy
Tags: array, string

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def isLongPressedName(self, name: str, typed: str) -> bool:
```

```
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def isLongPressedName(self, name, typed):
"""
:type name: str
:type typed: str
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Long Pressed Name
* Difficulty: Easy
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {string} name
* @param {string} typed
* @return {boolean}
*/
var isLongPressedName = function(name, typed) {

};
```

TypeScript Solution:

```
/**
* Problem: Long Pressed Name
* Difficulty: Easy
* Tags: array, string
*
```

```
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function isLongPressedName(name: string, typed: string): boolean {

};
```

C# Solution:

```
/*
* Problem: Long Pressed Name
* Difficulty: Easy
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public bool IsLongPressedName(string name, string typed) {

}
}
```

C Solution:

```
/*
* Problem: Long Pressed Name
* Difficulty: Easy
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

bool isLongPressedName(char* name, char* typed) {
```

```
}
```

Go Solution:

```
// Problem: Long Pressed Name
// Difficulty: Easy
// Tags: array, string
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func isLongPressedName(name string, typed string) bool {

}
```

Kotlin Solution:

```
class Solution {
fun isLongPressedName(name: String, typed: String): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func isLongPressedName(_ name: String, _ typed: String) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Long Pressed Name
// Difficulty: Easy
// Tags: array, string
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach
```

```
impl Solution {
pub fn is_long_pressed_name(name: String, typed: String) -> bool {

}
}
```

Ruby Solution:

```
# @param {String} name
# @param {String} typed
# @return {Boolean}
def is_long_pressed_name(name, typed)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $name
* @param String $typed
* @return Boolean
*/
function isLongPressedName($name, $typed) {

}
}
```

Dart Solution:

```
class Solution {
bool isLongPressedName(String name, String typed) {

}
}
```

Scala Solution:

```
object Solution {
def isLongPressedName(name: String, typed: String): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec is_long_pressed_name(name :: String.t, typed :: String.t) :: boolean
def is_long_pressed_name(name, typed) do

end
end
```

Erlang Solution:

```
-spec is_long_pressed_name(Name :: unicode:unicode_binary(), Typed ::
unicode:unicode_binary()) -> boolean().
is_long_pressed_name(Name, Typed) ->
.
```

Racket Solution:

```
(define/contract (is-long-pressed-name name typed)
(-> string? string? boolean?)
)
```