Problem 1935: Maximum Number of Words You Can Type

Problem Information

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

Problem Description

There is a malfunctioning keyboard where some letter keys do not work. All other keys on the keyboard work properly.

Given a string

text

of words separated by a single space (no leading or trailing spaces) and a string

brokenLetters

of all

distinct

letter keys that are broken, return

the

number of words

in

text

you can fully type using this keyboard

.

Example 1:

Input:

text = "hello world", brokenLetters = "ad"

Output:

1

Explanation:

We cannot type "world" because the 'd' key is broken.

Example 2:

Input:

text = "leet code", brokenLetters = "lt"

Output:

1

Explanation:

We cannot type "leet" because the 'l' and 't' keys are broken.

Example 3:

Input:

text = "leet code", brokenLetters = "e"

Output:

0

Explanation:

We cannot type either word because the 'e' key is broken.

Constraints:

1 <= text.length <= 10

4

0 <= brokenLetters.length <= 26

text

consists of words separated by a single space without any leading or trailing spaces.

Each word only consists of lowercase English letters.

brokenLetters

consists of

distinct

lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
int canBeTypedWords(string text, string brokenLetters) {

}
};
```

Java:

```java
class Solution {
public int canBeTypedWords(String text, String brokenLetters) {

}
}
```

Python3:

```python
class Solution:
def canBeTypedWords(self, text: str, brokenLetters: str) -> int:
```

Python:

```python
class Solution(object):
def canBeTypedWords(self, text, brokenLetters):
"""
:type text: str
:type brokenLetters: str
:rtype: int
"""
```

JavaScript:

```javascript
/**
* @param {string} text
* @param {string} brokenLetters
* @return {number}
*/
var canBeTypedWords = function(text, brokenLetters) {

};
```

TypeScript:

```typescript
function canBeTypedWords(text: string, brokenLetters: string): number {

};
```

C#:

```
}
```

```
public class Solution {
public int CanBeTypedWords(string text, string brokenLetters) {

}
}
```

C:

```
int canBeTypedWords(char* text, char* brokenLetters) {

}
```

Go:

```
func canBeTypedWords(text string, brokenLetters string) int {

}
```

Kotlin:

```
class Solution {
fun canBeTypedWords(text: String, brokenLetters: String): Int {

}
}
```

Swift:

```
class Solution {
func canBeTypedWords(_ text: String, _ brokenLetters: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn can_be_typed_words(text: String, broken_letters: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} text
# @param {String} broken_letters
# @return {Integer}
def can_be_typed_words(text, broken_letters)

end
```

PHP:

```
class Solution {

/**
* @param String $text
* @param String $brokenLetters
* @return Integer
*/
function canBeTypedWords($text, $brokenLetters) {

}
}
```

Dart:

```
class Solution {
int canBeTypedWords(String text, String brokenLetters) {

}
}
```

Scala:

```
object Solution {
def canBeTypedWords(text: String, brokenLetters: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec can_be_typed_words(text :: String.t, broken_letters :: String.t) ::
integer
def can_be_typed_words(text, broken_letters) do
```

```


end
end
```

Erlang:

```
-spec can_be_typed_words(Text :: unicode:unicode_binary(), BrokenLetters ::
unicode:unicode_binary()) -> integer().
can_be_typed_words(Text, BrokenLetters) ->
.
```

Racket:

```
(define/contract (can-be-typed-words text brokenLetters)
(-> string? string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Number of Words You Can Type
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
int canBeTypedWords(string text, string brokenLetters) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Number of Words You Can Type
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public int canBeTypedWords(String text, String brokenLetters) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Number of Words You Can Type
Difficulty: Easy
Tags: string, hash

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class Solution:
def canBeTypedWords(self, text: str, brokenLetters: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def canBeTypedWords(self, text, brokenLetters):
"""
:type text: str
:type brokenLetters: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Maximum Number of Words You Can Type
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {string} text
* @param {string} brokenLetters
* @return {number}
*/
var canBeTypedWords = function(text, brokenLetters) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Number of Words You Can Type
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function canBeTypedWords(text: string, brokenLetters: string): number {

};
```

C# Solution:

```
/*
* Problem: Maximum Number of Words You Can Type
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int CanBeTypedWords(string text, string brokenLetters) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Number of Words You Can Type
* Difficulty: Easy
* Tags: string, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

int canBeTypedWords(char* text, char* brokenLetters) {

}
```

Go Solution:

```
// Problem: Maximum Number of Words You Can Type
// Difficulty: Easy
// Tags: string, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func canBeTypedWords(text string, brokenLetters string) int {


}
```

Kotlin Solution:

```
class Solution {
fun canBeTypedWords(text: String, brokenLetters: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func canBeTypedWords(_ text: String, _ brokenLetters: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Maximum Number of Words You Can Type
// Difficulty: Easy
// Tags: string, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn can_be_typed_words(text: String, broken_letters: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} text
# @param {String} broken_letters
# @return {Integer}
def can_be_typed_words(text, broken_letters)
```

```
end
```

PHP Solution:

```
class Solution {

/**
* @param String $text
* @param String $brokenLetters
* @return Integer
*/
function canBeTypedWords($text, $brokenLetters) {

}
}
```

Dart Solution:

```
class Solution {
int canBeTypedWords(String text, String brokenLetters) {

}
}
```

Scala Solution:

```
object Solution {
def canBeTypedWords(text: String, brokenLetters: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec can_be_typed_words(text :: String.t, broken_letters :: String.t) ::
integer
def can_be_typed_words(text, broken_letters) do

end
end
```

Erlang Solution:

```
-spec can_be_typed_words(Text :: unicode:unicode_binary(), BrokenLetters ::
unicode:unicode_binary()) -> integer().
can_be_typed_words(Text, BrokenLetters) ->
.
```

Racket Solution:

```
(define/contract (can-be-typed-words text brokenLetters)
(-> string? string? exact-integer?)
)
```