Problem 3136: Valid Word

Problem Information

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

Problem Description

A word is considered

valid

if:

It contains a

minimum

of 3 characters.

It contains only digits (0-9), and English letters (uppercase and lowercase).

It includes

at least

one

vowel

.

It includes

'

at least

one

consonant

.

You are given a string

word

.

Return

true

if

word

is valid, otherwise, return

false

.

Notes:

'a'

,

'e'

,

'i'

,

'o'

,

'u'

, and their uppercases are

vowels

.

A

consonant

is an English letter that is not a vowel.

Example 1:

Input:

word = "234Adas"

Output:

true

Explanation:

This word satisfies the conditions.

Example 2:

Input:

word = "b3"

Output:

false

Explanation:

The length of this word is fewer than 3, and does not have a vowel.

Example 3:

Input:

word = "a3$e"

Output:

false

Explanation:

This word contains a

'$'

character and does not have a consonant.

Constraints:

1 <= word.length <= 20

word

consists of English uppercase and lowercase letters, digits,

'@'

,

'#'

, and

'$'

.

Code Snippets

C++:

```
class Solution {
public:
bool isValid(string word) {

}
};
```

Java:

```
class Solution {
public boolean isValid(String word) {

}
}
```

Python3:

```
class Solution:
def isValid(self, word: str) -> bool:
```

Python:

```
class Solution(object):
def isValid(self, word):
"""
:type word: str
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {string} word
* @return {boolean}
*/
var isValid = function(word) {

};
```

TypeScript:

```
function isValid(word: string): boolean {

};
```

C#:

```
public class Solution {
public bool IsValid(string word) {

}
}
```

C:

```
bool isValid(char* word) {

}
```

Go:

```
func isValid(word string) bool {

}
```

Kotlin:

```
class Solution {
fun isValid(word: String): Boolean {

}
}
```

Swift:

```
class Solution {
func isValid(_ word: String) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn is_valid(word: String) -> bool {

}
}
```

Ruby:

```
# @param {String} word
# @return {Boolean}
def is_valid(word)

end
```

PHP:

```
class Solution {

/**
* @param String $word
* @return Boolean
*/
function isValid($word) {

}
}
```

Dart:

```
class Solution {
bool isValid(String word) {

}


}
```

Scala:

```
object Solution {
def isValid(word: String): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec is_valid(word :: String.t) :: boolean
def is_valid(word) do

end
end
```

Erlang:

```
-spec is_valid(Word :: unicode:unicode_binary()) -> boolean().
is_valid(Word) ->
.
```

Racket:

```
(define/contract (is-valid word)
(-> string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Valid Word
* 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:
bool isValid(string word) {

}
};
```

Java Solution:

```
/**
* Problem: Valid Word
* 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 boolean isValid(String word) {

}
}
```

Python3 Solution:

```
"""
Problem: Valid Word
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 isValid(self, word: str) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def isValid(self, word):
"""
:type word: str
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Valid Word
* 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} word
* @return {boolean}
*/
var isValid = function(word) {

};
```

TypeScript Solution:

```
/**
* Problem: Valid Word
* 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 isValid(word: string): boolean {

};
```

C# Solution:

```
/*
* Problem: Valid Word
* 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 bool IsValid(string word) {

}
}
```

C Solution:

```
/*
* Problem: Valid Word
* 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
*/

bool isValid(char* word) {


}
```

Go Solution:

```
// Problem: Valid Word
// 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 isValid(word string) bool {

}
```

Kotlin Solution:

```
class Solution {
fun isValid(word: String): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func isValid(_ word: String) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Valid Word
// 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 is_valid(word: String) -> bool {

}
}
```

Ruby Solution:

```
# @param {String} word
# @return {Boolean}
def is_valid(word)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $word
* @return Boolean
*/
function isValid($word) {

}
}
```

Dart Solution:

```
class Solution {
bool isValid(String word) {

}
}
```

Scala Solution:

```
object Solution {
def isValid(word: String): Boolean = {
}
}
```

Elixir Solution:

```
defmodule Solution do
@spec is_valid(word :: String.t) :: boolean
def is_valid(word) do

end
end
```

Erlang Solution:

```
-spec is_valid(Word :: unicode:unicode_binary()) -> boolean().
is_valid(Word) ->
.
```

Racket Solution:

```
(define/contract (is-valid word)
(-> string? boolean?)
)
```