Problem 520: Detect Capital

Problem Information

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

Problem Description

We define the usage of capitals in a word to be right when one of the following cases holds:

All letters in this word are capitals, like

"USA"

.

All letters in this word are not capitals, like

"leetcode"

.

Only the first letter in this word is capital, like

"Google"

.

Given a string

word

, return

true

if the usage of capitals in it is right.

Example 1:

Input:

word = "USA"

Output:

true

Example 2:

Input:

word = "FlaG"

Output:

false

Constraints:

1 <= word.length <= 100

word

consists of lowercase and uppercase English letters.

Code Snippets

C++:

```
class Solution {
public:
```

```
bool detectCapitalUse(string word) {

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
bool detectCapitalUse(char* word) {

}
```

Go:

```
func detectCapitalUse(word string) bool {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

```
end
end
```

Erlang:

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

Racket:

```
(define/contract (detect-capital-use word)
(-> string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Detect Capital
* 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 detectCapitalUse(string word) {

}
};
```

Java Solution:

```
/**
* Problem: Detect Capital
```

```
* 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 detectCapitalUse(String word) {

}
}
```

Python3 Solution:

```
"""
Problem: Detect Capital
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 detectCapitalUse(self, word: str) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Detect Capital
* 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 detectCapitalUse = function(word) {

};
```

TypeScript Solution:

```
/**
* Problem: Detect Capital
* 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 detectCapitalUse(word: string): boolean {

};
```

C# Solution:

```
/*
* Problem: Detect Capital
* 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 DetectCapitalUse(string word) {

}
}
```

C Solution:

```
/*
* Problem: Detect Capital
* 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 detectCapitalUse(char* word) {

}
```

Go Solution:

```
// Problem: Detect Capital
// 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 detectCapitalUse(word string) bool {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Detect Capital
// 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 detect_capital_use(word: String) -> bool {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {
```

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (detect-capital-use word)
(-> string? boolean?)
)
```