Problem 1704: Determine if String Halves Are Alike

Problem Information

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

Problem Description

You are given a string

s

of even length. Split this string into two halves of equal lengths, and let

a

be the first half and

b

be the second half.

Two strings are

alike

if they have the same number of vowels (

'a'

,

'e'

,

'i'

,

'o'

,

'u'

,

'A'

,

'E'

,

'I'

,

'O'

,

'U'

). Notice that

s

contains uppercase and lowercase letters.

Return

true

if

a

and

b

are

alike

. Otherwise, return

false

.

Example 1:

Input:

s = "book"

Output:

true

Explanation:

a = "b

o

" and b = "

o

k". a has 1 vowel and b has 1 vowel. Therefore, they are alike.

Example 2:

Input:

s = "textbook"

Output:

false

Explanation:

a = "t

e

xt" and b = "b

oo

k". a has 1 vowel whereas b has 2. Therefore, they are not alike. Notice that the vowel o is counted twice.

Constraints:

2 <= s.length <= 1000

s.length

is even.

s

consists of

uppercase and lowercase

letters.

Code Snippets

C++:

```
class Solution {
public:
bool halvesAreAlike(string s) {

}
};
```

Java:

```
class Solution {
public boolean halvesAreAlike(String s) {

}
}
```

Python3:

```
class Solution:
def halvesAreAlike(self, s: str) -> bool:
```

Python:

```
class Solution(object):
def halvesAreAlike(self, s):
"""
:type s: str
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {string} s
* @return {boolean}
*/
var halvesAreAlike = function(s) {

};
```

TypeScript:

```
function halvesAreAlike(s: string): boolean {

};
```

C#:

```
public class Solution {
public bool HalvesAreAlike(string s) {

}
}
```

C:

```
bool halvesAreAlike(char* s) {

}
```

Go:

```
func halvesAreAlike(s string) bool {

}
```

Kotlin:

```
class Solution {
fun halvesAreAlike(s: String): Boolean {

}
}
```

Swift:

```
class Solution {
func halvesAreAlike(_ s: String) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn halves_are_alike(s: String) -> bool {

}
}
```

Ruby:

```
# @param {String} s
# @return {Boolean}
def halves_are_alike(s)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @return Boolean
*/
function halvesAreAlike($s) {

}
}
```

Dart:

```
class Solution {
bool halvesAreAlike(String s) {

}
}
```

Scala:

```
object Solution {
def halvesAreAlike(s: String): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec halves_are_alike(s :: String.t) :: boolean
def halves_are_alike(s) do

end
end
```

Erlang:

```
-spec halves_are_alike(S :: unicode:unicode_binary()) -> boolean().
halves_are_alike(S) ->
.
```

Racket:

```
(define/contract (halves-are-alike s)
(-> string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Determine if String Halves Are Alike
* 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 halvesAreAlike(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Determine if String Halves Are Alike
* 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 halvesAreAlike(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Determine if String Halves Are Alike
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 halvesAreAlike(self, s: str) -> bool:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

```
class Solution(object):
def halvesAreAlike(self, s):
"""
:type s: str
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Determine if String Halves Are Alike
* 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} s
* @return {boolean}
*/
var halvesAreAlike = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Determine if String Halves Are Alike
* 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 halvesAreAlike(s: string): boolean {

};
```

C# Solution:

```
/*
* Problem: Determine if String Halves Are Alike
* 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 HalvesAreAlike(string s) {

}
}
```

C Solution:

```
/*
* Problem: Determine if String Halves Are Alike
* 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 halvesAreAlike(char* s) {

}
```

Go Solution:

```
// Problem: Determine if String Halves Are Alike
// 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 halvesAreAlike(s string) bool {

}
```

Kotlin Solution:

```
class Solution {
fun halvesAreAlike(s: String): Boolean {

}
}
```

Swift Solution:

```
class Solution {
func halvesAreAlike(_ s: String) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Determine if String Halves Are Alike
// 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 halves_are_alike(s: String) -> bool {

}

}
```

Ruby Solution:

```
# @param {String} s
# @return {Boolean}
def halves_are_alike(s)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @return Boolean
*/
function halvesAreAlike($s) {

}
}
```

Dart Solution:

```
class Solution {
bool halvesAreAlike(String s) {

}
}
```

Scala Solution:

```
object Solution {
def halvesAreAlike(s: String): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec halves_are_alike(s :: String.t) :: boolean
def halves_are_alike(s) do

end
end
```

Erlang Solution:

```
-spec halves_are_alike(S :: unicode:unicode_binary()) -> boolean().
halves_are_alike(S) ->
.
```

Racket Solution:

```
(define/contract (halves-are-alike s)
(-> string? boolean?)
)
```