Problem 1880: Check if Word Equals Summation of Two Words

Problem Information

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

Problem Description

The

letter value

of a letter is its position in the alphabet

starting from 0

(i.e.

'a' -> 0

,

'b' -> 1

,

'c' -> 2

, etc.).

The

numerical value

of some string of lowercase English letters

s

is the

concatenation

of the

letter values

of each letter in

s

, which is then

converted

into an integer.

For example, if

s = "acb"

, we concatenate each letter's letter value, resulting in

"021"

. After converting it, we get

21

.

You are given three strings

firstWord

,

secondWord

, and

targetWord

, each consisting of lowercase English letters

'a'

through

'j'

inclusive

.

Return

true

if the

summation

of the

numerical values

of

firstWord

and

secondWord

equals the

numerical value

of

targetWord

, or

false

otherwise.

Example 1:

Input:

firstWord = "acb", secondWord = "cba", targetWord = "cdb"

Output:

true

Explanation:

The numerical value of firstWord is "acb" -> "021" -> 21. The numerical value of secondWord is "cba" -> "210" -> 210. The numerical value of targetWord is "cdb" -> "231" -> 231. We return true because 21 + 210 == 231.

Example 2:

Input:

firstWord = "aaa", secondWord = "a", targetWord = "aab"

Output:

false

Explanation:

The numerical value of firstWord is "aaa" -> "000" -> 0. The numerical value of secondWord is "a" -> "0" -> 0. The numerical value of targetWord is "aab" -> "001" -> 1. We return false because 0 + 0 != 1.

Example 3:

Input:

firstWord = "aaa", secondWord = "a", targetWord = "aaaa"

Output:

true

Explanation:

The numerical value of firstWord is "aaa" -> "000" -> 0. The numerical value of secondWord is "a" -> "0" -> 0. The numerical value of targetWord is "aaaa" -> "0000" -> 0. We return true because 0 + 0 == 0.

Constraints:

1 <= firstWord.length,

secondWord.length,

targetWord.length <= 8

firstWord

,

secondWord

, and

targetWord

consist of lowercase English letters from

'a'

to

'j'

inclusive

.

Code Snippets

C++:

```
class Solution {
public:
bool isSumEqual(string firstWord, string secondWord, string targetWord) {

}
};
```

Java:

```
class Solution {
public boolean isSumEqual(String firstWord, String secondWord, String
targetWord) {

}
}
```

Python3:

```
class Solution:
def isSumEqual(self, firstWord: str, secondWord: str, targetWord: str) ->
bool:
```

Python:

```
class Solution(object):
def isSumEqual(self, firstWord, secondWord, targetWord):
"""
:type firstWord: str
:type secondWord: str
:type targetWord: str
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {string} firstWord
* @param {string} secondWord
* @param {string} targetWord
* @return {boolean}
*/
var isSumEqual = function(firstWord, secondWord, targetWord) {

};
```

TypeScript:

```
function isSumEqual(firstWord: string, secondWord: string, targetWord:
string): boolean {

};
```

C#:

```
public class Solution {
public bool IsSumEqual(string firstWord, string secondWord, string
targetWord) {

}
}
```

C:

```
bool isSumEqual(char* firstWord, char* secondWord, char* targetWord) {
```

```
}
```

Go:

```
func isSumEqual(firstWord string, secondWord string, targetWord string) bool
{

}
```

Kotlin:

```
class Solution {
fun isSumEqual(firstWord: String, secondWord: String, targetWord: String):
Boolean {

}
}
```

Swift:

```
class Solution {
func isSumEqual(_ firstWord: String, _ secondWord: String, _ targetWord:
String) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn is_sum_equal(first_word: String, second_word: String, target_word:
String) -> bool {

}
}
```

Ruby:

```
# @param {String} first_word
# @param {String} second_word
# @param {String} target_word
# @return {Boolean}
```

```
def is_sum_equal(first_word, second_word, target_word)

end
```

PHP:

```
class Solution {

/**
* @param String $firstWord
* @param String $secondWord
* @param String $targetWord
* @return Boolean
*/
function isSumEqual($firstWord, $secondWord, $targetWord) {

}
}
```

Dart:

```
class Solution {
bool isSumEqual(String firstWord, String secondWord, String targetWord) {

}
}
```

Scala:

```
object Solution {
def isSumEqual(firstWord: String, secondWord: String, targetWord: String):
Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec is_sum_equal(first_word :: String.t, second_word :: String.t,
target_word :: String.t) :: boolean
def is_sum_equal(first_word, second_word, target_word) do
```

```
end
end
```

Erlang:

```
-spec is_sum_equal(FirstWord :: unicode:unicode_binary(), SecondWord ::
unicode:unicode_binary(), TargetWord :: unicode:unicode_binary()) ->
boolean().
is_sum_equal(FirstWord, SecondWord, TargetWord) ->
.
```

Racket:

```
(define/contract (is-sum-equal firstWord secondWord targetWord)
(-> string? string? string? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Check if Word Equals Summation of Two Words
* 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 isSumEqual(string firstWord, string secondWord, string targetWord) {

}
};
```

Java Solution:

```
/**
* Problem: Check if Word Equals Summation of Two Words
* 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 isSumEqual(String firstWord, String secondWord, String
targetWord) {

}
}
```

Python3 Solution:

```
"""
Problem: Check if Word Equals Summation of Two Words
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 isSumEqual(self, firstWord: str, secondWord: str, targetWord: str) ->
bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def isSumEqual(self, firstWord, secondWord, targetWord):
"""
:type firstWord: str
:type secondWord: str
```

```
:type targetWord: str
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Check if Word Equals Summation of Two Words
* 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} firstWord
* @param {string} secondWord
* @param {string} targetWord
* @return {boolean}
*/
var isSumEqual = function(firstWord, secondWord, targetWord) {

};
```

TypeScript Solution:

```
/**
* Problem: Check if Word Equals Summation of Two Words
* 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 isSumEqual(firstWord: string, secondWord: string, targetWord:
string): boolean {
```

```
};
```

C# Solution:

```
/*
* Problem: Check if Word Equals Summation of Two Words
* 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 IsSumEqual(string firstWord, string secondWord, string
targetWord) {

}
}
```

C Solution:

```
/*
* Problem: Check if Word Equals Summation of Two Words
* 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 isSumEqual(char* firstWord, char* secondWord, char* targetWord) {

}
```

Go Solution:

```
// Problem: Check if Word Equals Summation of Two Words
// 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 isSumEqual(firstWord string, secondWord string, targetWord string) bool
{

}
```

Kotlin Solution:

```
class Solution {
fun isSumEqual(firstWord: String, secondWord: String, targetWord: String):
Boolean {

}
}
```

Swift Solution:

```
class Solution {
func isSumEqual(_ firstWord: String, _ secondWord: String, _ targetWord:
String) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Check if Word Equals Summation of Two Words
// 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_sum_equal(first_word: String, second_word: String, target_word:
```

```
String) -> bool {

}
}
```

Ruby Solution:

```
# @param {String} first_word
# @param {String} second_word
# @param {String} target_word
# @return {Boolean}
def is_sum_equal(first_word, second_word, target_word)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $firstWord
* @param String $secondWord
* @param String $targetWord
* @return Boolean
*/
function isSumEqual($firstWord, $secondWord, $targetWord) {

}
}
```

Dart Solution:

```
class Solution {
bool isSumEqual(String firstWord, String secondWord, String targetWord) {

}
}
```

Scala Solution:

```
object Solution {
def isSumEqual(firstWord: String, secondWord: String, targetWord: String):
```

```
Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec is_sum_equal(first_word :: String.t, second_word :: String.t,
target_word :: String.t) :: boolean
def is_sum_equal(first_word, second_word, target_word) do

end
end
```

Erlang Solution:

```
-spec is_sum_equal(FirstWord :: unicode:unicode_binary(), SecondWord ::
unicode:unicode_binary(), TargetWord :: unicode:unicode_binary()) ->
boolean().
is_sum_equal(FirstWord, SecondWord, TargetWord) ->
.
```

Racket Solution:

```
(define/contract (is-sum-equal firstWord secondWord targetWord)
(-> string? string? string? boolean?)
)
```