Problem 537: Complex Number Multiplication

Problem Information

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

Problem Description

A

complex number

can be represented as a string on the form

"

real

+

imaginary

i"

where:

real

is the real part and is an integer in the range

[-100, 100]

.

imaginary

is the imaginary part and is an integer in the range

[-100, 100]

.

i

2

== -1

.

Given two complex numbers

num1

and

num2

as strings, return

a string of the complex number that represents their multiplications

.

Example 1:

Input:

num1 = "1+1i", num2 = "1+1i"

Output:

"0+2i"

Explanation:

(1 + i) * (1 + i) = 1 + i2 + 2 * i = 2i, and you need convert it to the form of 0+2i.

Example 2:

Input:

num1 = "1+-1i", num2 = "1+-1i"

Output:

"0+-2i"

Explanation:

(1 - i) * (1 - i) = 1 + i2 - 2 * i = -2i, and you need convert it to the form of 0+-2i.

Constraints:

num1

and

num2

are valid complex numbers.

Code Snippets

C++:

```
class Solution {
public:
string complexNumberMultiply(string num1, string num2) {

}
};
```

Java:

```
class Solution {
public String complexNumberMultiply(String num1, String num2) {

}
}
```

Python3:

```
class Solution:
def complexNumberMultiply(self, num1: str, num2: str) -> str:
```

Python:

```
class Solution(object):
def complexNumberMultiply(self, num1, num2):
"""
:type num1: str
:type num2: str
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} num1
* @param {string} num2
* @return {string}
*/
var complexNumberMultiply = function(num1, num2) {

};
```

TypeScript:

```
function complexNumberMultiply(num1: string, num2: string): string {

};
```

C#:

```
public class Solution {
public string ComplexNumberMultiply(string num1, string num2) {

}
}
```

C:

```
char* complexNumberMultiply(char* num1, char* num2) {

}
```

Go:

```
func complexNumberMultiply(num1 string, num2 string) string {

}
```

Kotlin:

```
class Solution {
fun complexNumberMultiply(num1: String, num2: String): String {

}
}
```

Swift:

```
class Solution {
func complexNumberMultiply(_ num1: String, _ num2: String) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn complex_number_multiply(num1: String, num2: String) -> String {

}
}
```

Ruby:

```
# @param {String} num1
# @param {String} num2
# @return {String}
def complex_number_multiply(num1, num2)

end
```

PHP:

```
class Solution {

/**
* @param String $num1
* @param String $num2
* @return String
*/
function complexNumberMultiply($num1, $num2) {

}
}
```

Dart:

```
class Solution {
String complexNumberMultiply(String num1, String num2) {

}
}
```

Scala:

```
object Solution {
def complexNumberMultiply(num1: String, num2: String): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec complex_number_multiply(num1 :: String.t, num2 :: String.t) :: String.t
def complex_number_multiply(num1, num2) do
```

```
end
end
```

Erlang:

```
-spec complex_number_multiply(Num1 :: unicode:unicode_binary(), Num2 ::
unicode:unicode_binary()) -> unicode:unicode_binary().
complex_number_multiply(Num1, Num2) ->
.
```

Racket:

```
(define/contract (complex-number-multiply num1 num2)
(-> string? string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Complex Number Multiplication
* Difficulty: Medium
* Tags: string, math
*
* 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:
string complexNumberMultiply(string num1, string num2) {

}
};
```

Java Solution:

```
/**
* Problem: Complex Number Multiplication
* Difficulty: Medium
* Tags: string, math
*
* 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 String complexNumberMultiply(String num1, String num2) {

}
}
```

Python3 Solution:

```
"""
Problem: Complex Number Multiplication
Difficulty: Medium
Tags: string, math

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 complexNumberMultiply(self, num1: str, num2: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def complexNumberMultiply(self, num1, num2):
"""
:type num1: str
:type num2: str
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Complex Number Multiplication
* Difficulty: Medium
* Tags: string, math
*
* 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} num1
* @param {string} num2
* @return {string}
*/
var complexNumberMultiply = function(num1, num2) {

};
```

TypeScript Solution:

```
/**
* Problem: Complex Number Multiplication
* Difficulty: Medium
* Tags: string, math
*
* 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 complexNumberMultiply(num1: string, num2: string): string {

};
```

C# Solution:

```
/*
* Problem: Complex Number Multiplication
* Difficulty: Medium
```

```
* Tags: string, math
*
* 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 string ComplexNumberMultiply(string num1, string num2) {

}
}
```

C Solution:

```
/*
* Problem: Complex Number Multiplication
* Difficulty: Medium
* Tags: string, math
*
* 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
*/

char* complexNumberMultiply(char* num1, char* num2) {

}
```

Go Solution:

```
// Problem: Complex Number Multiplication
// Difficulty: Medium
// Tags: string, math
//
// 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 complexNumberMultiply(num1 string, num2 string) string {
```

```
}
```

Kotlin Solution:

```
class Solution {
fun complexNumberMultiply(num1: String, num2: String): String {

}
}
```

Swift Solution:

```
class Solution {
func complexNumberMultiply(_ num1: String, _ num2: String) -> String {

}
}
```

Rust Solution:

```
// Problem: Complex Number Multiplication
// Difficulty: Medium
// Tags: string, math
//
// 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 complex_number_multiply(num1: String, num2: String) -> String {

}
}
```

Ruby Solution:

```
# @param {String} num1
# @param {String} num2
# @return {String}
def complex_number_multiply(num1, num2)
```

```
end
```

PHP Solution:

```
class Solution {

/**
* @param String $num1
* @param String $num2
* @return String
*/
function complexNumberMultiply($num1, $num2) {

}
}
```

Dart Solution:

```
class Solution {
String complexNumberMultiply(String num1, String num2) {

}
}
```

Scala Solution:

```
object Solution {
def complexNumberMultiply(num1: String, num2: String): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec complex_number_multiply(num1 :: String.t, num2 :: String.t) :: String.t
def complex_number_multiply(num1, num2) do

end
end
```

Erlang Solution:

```
-spec complex_number_multiply(Num1 :: unicode:unicode_binary(), Num2 ::
unicode:unicode_binary()) -> unicode:unicode_binary().
complex_number_multiply(Num1, Num2) ->
.
```

Racket Solution:

```
(define/contract (complex-number-multiply num1 num2)
(-> string? string? string?)
)
```