Problem 592: Fraction Addition and Subtraction

Problem Information

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

Problem Description

Given a string

expression

representing an expression of fraction addition and subtraction, return the calculation result in string format.

The final result should be an

irreducible fraction

. If your final result is an integer, change it to the format of a fraction that has a denominator

1

. So in this case,

2

should be converted to

2/1

.

Example 1:

Input:

expression = "-1/2+1/2"

Output:

"0/1"

Example 2:

Input:

expression = "-1/2+1/2+1/3"

Output:

"1/3"

Example 3:

Input:

expression = "1/3-1/2"

Output:

"-1/6"

Constraints:

The input string only contains

'0'

to

'9'

,

'/'

,

'+'

and

'-'

. So does the output.

Each fraction (input and output) has the format

±numerator/denominator

. If the first input fraction or the output is positive, then

'+'

will be omitted.

The input only contains valid

irreducible fractions

, where the

numerator

and

denominator

of each fraction will always be in the range

[1, 10]

. If the denominator is

1

, it means this fraction is actually an integer in a fraction format defined above.

The number of given fractions will be in the range

[1, 10]

.

The numerator and denominator of the

final result

are guaranteed to be valid and in the range of

32-bit

int.

Code Snippets

C++:

```
class Solution {
public:
string fractionAddition(string expression) {

}
};
```

Java:

```
class Solution {
public String fractionAddition(String expression) {

}
}
```

Python3:

```
class Solution:
def fractionAddition(self, expression: str) -> str:
```

Python:

```
class Solution(object):
def fractionAddition(self, expression):
"""
:type expression: str
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} expression
* @return {string}
*/
var fractionAddition = function(expression) {

};
```

TypeScript:

```
function fractionAddition(expression: string): string {

};
```

C#:

```
public class Solution {
public string FractionAddition(string expression) {

}
}
```

C:

```
char* fractionAddition(char* expression) {

}
```

Go:

```
func fractionAddition(expression string) string {

}
```

Kotlin:

```
class Solution {
fun fractionAddition(expression: String): String {

}
}
```

Swift:

```
class Solution {
func fractionAddition(_ expression: String) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn fraction_addition(expression: String) -> String {

}
}
```

Ruby:

```
# @param {String} expression
# @return {String}
def fraction_addition(expression)

end
```

PHP:

```
class Solution {

/**
* @param String $expression
* @return String
*/
function fractionAddition($expression) {

}
}
```

Dart:

```
class Solution {
String fractionAddition(String expression) {

}
}
```

Scala:

```
object Solution {
def fractionAddition(expression: String): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec fraction_addition(expression :: String.t) :: String.t
def fraction_addition(expression) do

end
end
```

Erlang:

```
-spec fraction_addition(Expression :: unicode:unicode_binary()) ->
unicode:unicode_binary().
fraction_addition(Expression) ->
.
```

Racket:

```
(define/contract (fraction-addition expression)
(-> string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Fraction Addition and Subtraction
* 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 fractionAddition(string expression) {

}
};
```

Java Solution:

```
/**
* Problem: Fraction Addition and Subtraction
* 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 fractionAddition(String expression) {
}
}
```

Python3 Solution:

```
"""
Problem: Fraction Addition and Subtraction
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 fractionAddition(self, expression: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def fractionAddition(self, expression):
"""
:type expression: str
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Fraction Addition and Subtraction
* 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} expression
* @return {string}
*/
var fractionAddition = function(expression) {

};
```

TypeScript Solution:

```
/**
* Problem: Fraction Addition and Subtraction
* 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 fractionAddition(expression: string): string {

};
```

C# Solution:

```
/*
* Problem: Fraction Addition and Subtraction
* 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 FractionAddition(string expression) {

}
}
```

C Solution:

```
/*
* Problem: Fraction Addition and Subtraction
* 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* fractionAddition(char* expression) {

}
```

Go Solution:

```
// Problem: Fraction Addition and Subtraction
// 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 fractionAddition(expression string) string {

}
```

Kotlin Solution:

```
class Solution {
fun fractionAddition(expression: String): String {

}
}
```

Swift Solution:

```
class Solution {
func fractionAddition(_ expression: String) -> String {

}
}
```

Rust Solution:

```
// Problem: Fraction Addition and Subtraction
// 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 fraction_addition(expression: String) -> String {

}
}
```

Ruby Solution:

```
# @param {String} expression
# @return {String}
def fraction_addition(expression)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $expression
* @return String
*/
function fractionAddition($expression) {

}
}
```

Dart Solution:

```
class Solution {
String fractionAddition(String expression) {

}
}
```

Scala Solution:

```
object Solution {
def fractionAddition(expression: String): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec fraction_addition(expression :: String.t) :: String.t
def fraction_addition(expression) do

end
end
```

Erlang Solution:

```
-spec fraction_addition(Expression :: unicode:unicode_binary()) ->
unicode:unicode_binary().
fraction_addition(Expression) ->
.
```

Racket Solution:

```
(define/contract (fraction-addition expression)
(-> string? string?)
)
```