Problem 726: Number of Atoms

Problem Information

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

Problem Description

Given a string

formula

representing a chemical formula, return

the count of each atom

.

The atomic element always starts with an uppercase character, then zero or more lowercase letters, representing the name.

One or more digits representing that element's count may follow if the count is greater than

1

. If the count is

1

, no digits will follow.

For example,

"H2O"

and

"H2O2"

are possible, but

"H1O2"

is impossible.

Two formulas are concatenated together to produce another formula.

For example,

"H2O2He3Mg4"

is also a formula.

A formula placed in parentheses, and a count (optionally added) is also a formula.

For example,

"(H2O2)"

and

"(H2O2)3"

are formulas.

Return the count of all elements as a string in the following form: the first name (in sorted order), followed by its count (if that count is more than

1

), followed by the second name (in sorted order), followed by its count (if that count is more than

1

), and so on.

The test cases are generated so that all the values in the output fit in a

32-bit

integer.

Example 1:

Input:

formula = "H2O"

Output:

"H2O"

Explanation:

The count of elements are {'H': 2, 'O': 1}.

Example 2:

Input:

formula = "Mg(OH)2"

Output:

"H2MgO2"

Explanation:

The count of elements are {'H': 2, 'Mg': 1, 'O': 2}.

Example 3:

Input:

formula = "K4(ON(SO3)2)2"

Output:

"K4N2O14S4"

Explanation:

The count of elements are {'K': 4, 'N': 2, 'O': 14, 'S': 4}.

Constraints:

1 <= formula.length <= 1000

formula

consists of English letters, digits,

'('

, and

')'

.

formula

is always valid.

Code Snippets

C++:

```
class Solution {
public:
```

```
string countOfAtoms(string formula) {

}
};
```

Java:

```
class Solution {
public String countOfAtoms(String formula) {

}
}
```

Python3:

```
class Solution:
def countOfAtoms(self, formula: str) -> str:
```

Python:

```
class Solution(object):
def countOfAtoms(self, formula):
"""
:type formula: str
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} formula
* @return {string}
*/
var countOfAtoms = function(formula) {

};
```

TypeScript:

```
function countOfAtoms(formula: string): string {

};
```

C#:

```
public class Solution {
public string CountOfAtoms(string formula) {

}
}
```

C:

```
char* countOfAtoms(char* formula) {

}
```

Go:

```
func countOfAtoms(formula string) string {

}
```

Kotlin:

```
class Solution {
fun countOfAtoms(formula: String): String {

}
}
```

Swift:

```
class Solution {
func countOfAtoms(_ formula: String) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn count_of_atoms(formula: String) -> String {

}
}
```

Ruby:

```
# @param {String} formula
# @return {String}
def count_of_atoms(formula)

end
```

PHP:

```
class Solution {

/**
* @param String $formula
* @return String
*/
function countOfAtoms($formula) {

}
}
```

Dart:

```
class Solution {
String countOfAtoms(String formula) {

}
}
```

Scala:

```
object Solution {
def countOfAtoms(formula: String): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec count_of_atoms(formula :: String.t) :: String.t
def count_of_atoms(formula) do
```

```
end
end
```

Erlang:

```
-spec count_of_atoms(Formula :: unicode:unicode_binary()) ->
unicode:unicode_binary().
count_of_atoms(Formula) ->
.
```

Racket:

```
(define/contract (count-of-atoms formula)
(-> string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Atoms
* Difficulty: Hard
* Tags: string, hash, sort, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
string countOfAtoms(string formula) {

}
};
```

Java Solution:

```
/**
* Problem: Number of Atoms
* Difficulty: Hard
* Tags: string, hash, sort, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public String countOfAtoms(String formula) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Atoms
Difficulty: Hard
Tags: string, hash, sort, stack

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class Solution:
def countOfAtoms(self, formula: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def countOfAtoms(self, formula):
"""
:type formula: str
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Number of Atoms
* Difficulty: Hard
* Tags: string, hash, sort, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {string} formula
* @return {string}
*/
var countOfAtoms = function(formula) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Atoms
* Difficulty: Hard
* Tags: string, hash, sort, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function countOfAtoms(formula: string): string {

};
```

C# Solution:

```
/*
* Problem: Number of Atoms
* Difficulty: Hard
* Tags: string, hash, sort, stack
*
```

```
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public string CountOfAtoms(string formula) {

}
}
```

C Solution:

```
/*
* Problem: Number of Atoms
* Difficulty: Hard
* Tags: string, hash, sort, stack
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

char* countOfAtoms(char* formula) {

}
```

Go Solution:

```
// Problem: Number of Atoms
// Difficulty: Hard
// Tags: string, hash, sort, stack
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func countOfAtoms(formula string) string {

}
```

Kotlin Solution:

```
class Solution {
fun countOfAtoms(formula: String): String {

}
}
```

Swift Solution:

```
class Solution {
func countOfAtoms(_ formula: String) -> String {

}
}
```

Rust Solution:

```
// Problem: Number of Atoms
// Difficulty: Hard
// Tags: string, hash, sort, stack
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn count_of_atoms(formula: String) -> String {

}
}
```

Ruby Solution:

```
# @param {String} formula
# @return {String}
def count_of_atoms(formula)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $formula
* @return String
*/
function countOfAtoms($formula) {

}
}
```

Dart Solution:

```
class Solution {
String countOfAtoms(String formula) {

}
}
```

Scala Solution:

```
object Solution {
def countOfAtoms(formula: String): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec count_of_atoms(formula :: String.t) :: String.t
def count_of_atoms(formula) do

end
end
```

Erlang Solution:

```
-spec count_of_atoms(Formula :: unicode:unicode_binary()) ->
unicode:unicode_binary().
count_of_atoms(Formula) ->
.
```

Racket Solution:

```
(define/contract (count-of-atoms formula)
(-> string? string?)
)
```