Problem 1678: Goal Parser Interpretation

Problem Information

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

Problem Description

You own a

Goal Parser

that can interpret a string

command

. The

command

consists of an alphabet of

"G"

,

"()"

and/or

"(al)"

in some order. The Goal Parser will interpret

"G"

as the string

"G"

,

"()"

as the string

"o"

, and

"(al)"

as the string

"al"

. The interpreted strings are then concatenated in the original order.

Given the string

command

, return

the

Goal Parser

's interpretation of

command

.

Example 1:

Input:

command = "G()(al)"

Output:

"Goal"

Explanation:

The Goal Parser interprets the command as follows: G -> G () -> o (al) -> al The final concatenated result is "Goal".

Example 2:

Input:

command = "G()()()()(al)"

Output:

"Gooooal"

Example 3:

Input:

command = "(al)G(al)()()G"

Output:

"alGalooG"

Constraints:

1 <= command.length <= 100

command

consists of

"G"

,

"()"

, and/or

"(al)"

in some order.

Code Snippets

C++:

```
class Solution {
public:
string interpret(string command) {

}
};
```

Java:

```
class Solution {
public String interpret(String command) {

}
}
```

Python3:

```
class Solution:
def interpret(self, command: str) -> str:
```

Python:

```
class Solution(object):
def interpret(self, command):
"""
:type command: str
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} command
* @return {string}
*/
var interpret = function(command) {

};
```

TypeScript:

```
function interpret(command: string): string {

};
```

C#:

```
public class Solution {
public string Interpret(string command) {

}
}
```

C:

```


char * interpret(char * command){

}
```

Go:

```
func interpret(command string) string {

}
```

Kotlin:

```
class Solution {
fun interpret(command: String): String {

}
}
```

Swift:

```
class Solution {
func interpret(_ command: String) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn interpret(command: String) -> String {

}
}
```

Ruby:

```
# @param {String} command
# @return {String}
def interpret(command)

end
```

PHP:

```
class Solution {

/**
* @param String $command
* @return String
```

```
*/
function interpret($command) {

}
}
```

Scala:

```
object Solution {
def interpret(command: String): String = {

}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Goal Parser Interpretation
* 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:
string interpret(string command) {

}
};
```

Java Solution:

```
/**
* Problem: Goal Parser Interpretation
* 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 String interpret(String command) {

}
}
```

Python3 Solution:

```
"""
Problem: Goal Parser Interpretation
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 interpret(self, command: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def interpret(self, command):
"""
:type command: str
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Goal Parser Interpretation
* 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} command
* @return {string}
*/
var interpret = function(command) {

};
```

TypeScript Solution:

```
/**
* Problem: Goal Parser Interpretation
* 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 interpret(command: string): string {

};
```

C# Solution:

```
/*
* Problem: Goal Parser Interpretation
* 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 string Interpret(string command) {

}
}
```

C Solution:

```
/*
* Problem: Goal Parser Interpretation
* 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
*/



char * interpret(char * command){

}
```

Go Solution:

```
// Problem: Goal Parser Interpretation
// 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 interpret(command string) string {

}
```

Kotlin Solution:

```
class Solution {
fun interpret(command: String): String {

}
}
```

Swift Solution:

```
class Solution {
func interpret(_ command: String) -> String {

}
}
```

Rust Solution:

```
// Problem: Goal Parser Interpretation
// 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 interpret(command: String) -> String {

}
}
```

Ruby Solution:

```
# @param {String} command
# @return {String}
def interpret(command)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $command
* @return String
*/
function interpret($command) {

}
}
```

Scala Solution:

```
object Solution {
def interpret(command: String): String = {

}
}
```