Problem 824: Goat Latin

Problem Information

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

Problem Description

You are given a string

sentence

that consist of words separated by spaces. Each word consists of lowercase and uppercase letters only.

We would like to convert the sentence to "Goat Latin" (a made-up language similar to Pig Latin.) The rules of Goat Latin are as follows:

If a word begins with a vowel (

'a'

,

'e'

,

'i'

,

'o'

, or

'u'

), append

"ma"

to the end of the word.

For example, the word

"apple"

becomes

"applema"

.

If a word begins with a consonant (i.e., not a vowel), remove the first letter and append it to the end, then add

"ma"

.

For example, the word

"goat"

becomes

"oatgma"

.

Add one letter

'a'

to the end of each word per its word index in the sentence, starting with

1

.

For example, the first word gets

"a"

added to the end, the second word gets

"aa"

added to the end, and so on.

Return

the final sentence representing the conversion from sentence to Goat Latin

.

Example 1:

Input:

sentence = "I speak Goat Latin"

Output:

"Imaa peaksmaaa oatGmaaaa atinLmaaaaa"

Example 2:

Input:

sentence = "The quick brown fox jumped over the lazy dog"

Output:

"heTmaa uickqmaaa rownbmaaaa oxfmaaaaa umpedjmaaaaaa overmaaaaaaa hetmaaaaaaaa azylmaaaaaaaaa ogdmaaaaaaaaaa"

Constraints:

1 <= sentence.length <= 150

sentence

consists of English letters and spaces.

sentence

has no leading or trailing spaces.

All the words in

sentence

are separated by a single space.

Code Snippets

C++:

```
class Solution {
public:
string toGoatLatin(string sentence) {

}
};
```

Java:

```
class Solution {
public String toGoatLatin(String sentence) {
```

```
}
}
```

Python3:

```
class Solution:
def toGoatLatin(self, sentence: str) -> str:
```

Python:

```
class Solution(object):
def toGoatLatin(self, sentence):
"""
:type sentence: str
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} sentence
* @return {string}
*/
var toGoatLatin = function(sentence) {

};
```

TypeScript:

```
function toGoatLatin(sentence: string): string {

};
```

C#:

```
public class Solution {
public string ToGoatLatin(string sentence) {

}
}
```

C:

```
char* toGoatLatin(char* sentence) {

}
```

Go:

```
func toGoatLatin(sentence string) string {

}
```

Kotlin:

```
class Solution {
fun toGoatLatin(sentence: String): String {

}
}
```

Swift:

```
class Solution {
func toGoatLatin(_ sentence: String) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn to_goat_latin(sentence: String) -> String {

}
}
```

Ruby:

```
# @param {String} sentence
# @return {String}
def to_goat_latin(sentence)

end
```

PHP:

```
class Solution {

/**
* @param String $sentence
* @return String
*/
function toGoatLatin($sentence) {

}
}
```

Dart:

```
class Solution {
String toGoatLatin(String sentence) {

}
}
```

Scala:

```
object Solution {
def toGoatLatin(sentence: String): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec to_goat_latin(sentence :: String.t) :: String.t
def to_goat_latin(sentence) do

end
end
```

Erlang:

```
-spec to_goat_latin(Sentence :: unicode:unicode_binary()) ->
unicode:unicode_binary().
to_goat_latin(Sentence) ->
.
```

Racket:

```
(define/contract (to-goat-latin sentence)
(-> string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Goat Latin
* 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 toGoatLatin(string sentence) {

}
};
```

Java Solution:

```
/**
* Problem: Goat Latin
* 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 toGoatLatin(String sentence) {
```

```
}
}
```

Python3 Solution:

```
"""
Problem: Goat Latin
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 toGoatLatin(self, sentence: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def toGoatLatin(self, sentence):
"""
:type sentence: str
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Goat Latin
* 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} sentence
* @return {string}
*/
var toGoatLatin = function(sentence) {

};
```

TypeScript Solution:

```
/**
* Problem: Goat Latin
* 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 toGoatLatin(sentence: string): string {

};
```

C# Solution:

```
/*
* Problem: Goat Latin
* 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 ToGoatLatin(string sentence) {

}
```

```
}
```

C Solution:

```
/*
* Problem: Goat Latin
* 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* toGoatLatin(char* sentence) {

}
```

Go Solution:

```
// Problem: Goat Latin
// 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 toGoatLatin(sentence string) string {

}
```

Kotlin Solution:

```
class Solution {
fun toGoatLatin(sentence: String): String {

}
}
```

Swift Solution:

```
class Solution {
func toGoatLatin(_ sentence: String) -> String {

}
}
```

Rust Solution:

```
// Problem: Goat Latin
// 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 to_goat_latin(sentence: String) -> String {

}
}
```

Ruby Solution:

```
# @param {String} sentence
# @return {String}
def to_goat_latin(sentence)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $sentence
* @return String
*/
function toGoatLatin($sentence) {

}
}
```

Dart Solution:

```
class Solution {
String toGoatLatin(String sentence) {

}
}
```

Scala Solution:

```
object Solution {
def toGoatLatin(sentence: String): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec to_goat_latin(sentence :: String.t) :: String.t
def to_goat_latin(sentence) do

end
end
```

Erlang Solution:

```
-spec to_goat_latin(Sentence :: unicode:unicode_binary()) ->
unicode:unicode_binary().
to_goat_latin(Sentence) ->
.
```

Racket Solution:

```
(define/contract (to-goat-latin sentence)
(-> string? string?)
)
```