Problem 1410: HTML Entity Parser

Problem Information

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

Problem Description

HTML entity parser

is the parser that takes HTML code as input and replace all the entities of the special characters by the characters itself.

The special characters and their entities for HTML are:

Quotation Mark:

the entity is

"

and symbol character is

"

.

Single Quote Mark:

the entity is

'

and symbol character is

'

.

Ampersand:

the entity is

&

and symbol character is

&

.

Greater Than Sign:

the entity is

>

and symbol character is

>

.

Less Than Sign:

the entity is

<

and symbol character is

<

.

Slash:

the entity is

⁄

and symbol character is

/

.

Given the input

text

string to the HTML parser, you have to implement the entity parser.

Return

the text after replacing the entities by the special characters

.

Example 1:

Input:

text = "& is an HTML entity but &ambassador; is not."

Output:

"& is an HTML entity but &ambassador; is not."

Explanation:

The parser will replace the & entity by &

Example 2:

Input:

text = "and I quote: "...""

Output:

"and I quote: \"...\""

Constraints:

1 <= text.length <= 10

5

The string may contain any possible characters out of all the 256 ASCII characters.

Code Snippets

C++:

```
class Solution {
public:
string entityParser(string text) {

}
};
```

Java:

```
class Solution {
public String entityParser(String text) {

}
}
```

Python3:

```
class Solution:
def entityParser(self, text: str) -> str:
```

Python:

```
class Solution(object):
def entityParser(self, text):
"""
:type text: str
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} text
* @return {string}
*/
var entityParser = function(text) {

};
```

TypeScript:

```
function entityParser(text: string): string {

};
```

C#:

```
public class Solution {
public string EntityParser(string text) {

}
}
```

C:

```
char* entityParser(char* text) {

}
```

Go:

```
func entityParser(text string) string {

}
```

Kotlin:

```
class Solution {
fun entityParser(text: String): String {

}
}
```

Swift:

```
class Solution {
func entityParser(_ text: String) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn entity_parser(text: String) -> String {

}
}
```

Ruby:

```
# @param {String} text
# @return {String}
def entity_parser(text)

end
```

PHP:

```
class Solution {

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

```
*/
function entityParser($text) {

}
}
```

Dart:

```
class Solution {
String entityParser(String text) {

}
}
```

Scala:

```
object Solution {
def entityParser(text: String): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec entity_parser(text :: String.t) :: String.t
def entity_parser(text) do

end
end
```

Erlang:

```
-spec entity_parser(Text :: unicode:unicode_binary()) ->
unicode:unicode_binary().
entity_parser(Text) ->
.
```

Racket:

```
(define/contract (entity-parser text)
(-> string? string?)
```

```
)
```

Solutions

C++ Solution:

```
/*
* Problem: HTML Entity Parser
* Difficulty: Medium
* Tags: string, hash
*
* 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 entityParser(string text) {

}
};
```

Java Solution:

```
/**
* Problem: HTML Entity Parser
* Difficulty: Medium
* Tags: string, hash
*
* 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 entityParser(String text) {

}
}
```

Python3 Solution:

```
"""
Problem: HTML Entity Parser
Difficulty: Medium
Tags: string, hash

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 entityParser(self, text: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def entityParser(self, text):
"""
:type text: str
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: HTML Entity Parser
* Difficulty: Medium
* Tags: string, hash
*
* 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} text
* @return {string}
*/
```

```
var entityParser = function(text) {

};
```

TypeScript Solution:

```
/**
* Problem: HTML Entity Parser
* Difficulty: Medium
* Tags: string, hash
*
* 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 entityParser(text: string): string {

};
```

C# Solution:

```
/*
* Problem: HTML Entity Parser
* Difficulty: Medium
* Tags: string, hash
*
* 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 EntityParser(string text) {

}
}
```

C Solution:

```
/*
* Problem: HTML Entity Parser
* Difficulty: Medium
* Tags: string, hash
*
* 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* entityParser(char* text) {

}
```

Go Solution:

```
// Problem: HTML Entity Parser
// Difficulty: Medium
// Tags: string, hash
//
// 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 entityParser(text string) string {

}
```

Kotlin Solution:

```
class Solution {
fun entityParser(text: String): String {

}
}
```

Swift Solution:

```
class Solution {
func entityParser(_ text: String) -> String {

}
```

```
}
```

Rust Solution:

```
// Problem: HTML Entity Parser
// Difficulty: Medium
// Tags: string, hash
//
// 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 entity_parser(text: String) -> String {

}
}
```

Ruby Solution:

```
# @param {String} text
# @return {String}
def entity_parser(text)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $text
* @return String
*/
function entityParser($text) {

}
}
```

Dart Solution:

```
class Solution {
String entityParser(String text) {

}
}
```

Scala Solution:

```
object Solution {
def entityParser(text: String): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec entity_parser(text :: String.t) :: String.t
def entity_parser(text) do

end
end
```

Erlang Solution:

```
-spec entity_parser(Text :: unicode:unicode_binary()) ->
unicode:unicode_binary().
entity_parser(Text) ->
.
```

Racket Solution:

```
(define/contract (entity-parser text)
(-> string? string?)
)
```