Problem 2309: Greatest English Letter in Upper and Lower Case

Problem Information

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

Problem Description

Given a string of English letters

s

, return

the

greatest

English letter which occurs as

both

a lowercase and uppercase letter in

s

. The returned letter should be in

uppercase

. If no such letter exists, return

an empty string

.

An English letter

b

is

greater

than another letter

a

if

b

appears

after

a

in the English alphabet.

Example 1:

Input:

s = "l

Ee

TcOd

E

"

Output:

"E"

Explanation:

The letter 'E' is the only letter to appear in both lower and upper case.

Example 2:

Input:

s = "a

rR

AzFif"

Output:

"R"

Explanation:

The letter 'R' is the greatest letter to appear in both lower and upper case. Note that 'A' and 'F' also appear in both lower and upper case, but 'R' is greater than 'F' or 'A'.

Example 3:

Input:

s = "AbCdEfGhIjK"

Output:

""

Explanation:

There is no letter that appears in both lower and upper case.

Constraints:

1 <= s.length <= 1000

s

consists of lowercase and uppercase English letters.

Code Snippets

C++:

```
class Solution {
public:
string greatestLetter(string s) {

}
};
```

Java:

```
class Solution {
public String greatestLetter(String s) {

}
}
```

Python3:

```
class Solution:
def greatestLetter(self, s: str) -> str:
```

Python:

```
class Solution(object):
def greatestLetter(self, s):
```

```
"""
:type s: str
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} s
* @return {string}
*/
var greatestLetter = function(s) {

};
```

TypeScript:

```
function greatestLetter(s: string): string {

};
```

C#:

```
public class Solution {
public string GreatestLetter(string s) {

}
}
```

C:

```
char* greatestLetter(char* s) {

}
```

Go:

```
func greatestLetter(s string) string {

}
```

Kotlin:

```
class Solution {
fun greatestLetter(s: String): String {

}
}
```

Swift:

```
class Solution {
func greatestLetter(_ s: String) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn greatest_letter(s: String) -> String {

}
}
```

Ruby:

```
# @param {String} s
# @return {String}
def greatest_letter(s)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @return String
*/
function greatestLetter($s) {

}
}
```

Dart:

```
class Solution {
String greatestLetter(String s) {

}
}
```

Scala:

```
object Solution {
def greatestLetter(s: String): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec greatest_letter(s :: String.t) :: String.t
def greatest_letter(s) do

end
end
```

Erlang:

```
-spec greatest_letter(S :: unicode:unicode_binary()) ->
unicode:unicode_binary().
greatest_letter(S) ->
.
```

Racket:

```
(define/contract (greatest-letter s)
(-> string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Greatest English Letter in Upper and Lower Case
* Difficulty: Easy
* 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 greatestLetter(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Greatest English Letter in Upper and Lower Case
* Difficulty: Easy
* 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 greatestLetter(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Greatest English Letter in Upper and Lower Case
Difficulty: Easy
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 greatestLetter(self, s: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def greatestLetter(self, s):
"""
:type s: str
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Greatest English Letter in Upper and Lower Case
* Difficulty: Easy
* 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} s
* @return {string}
*/
var greatestLetter = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Greatest English Letter in Upper and Lower Case
* Difficulty: Easy
* 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 greatestLetter(s: string): string {

};
```

C# Solution:

```
/*
* Problem: Greatest English Letter in Upper and Lower Case
* Difficulty: Easy
* 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 GreatestLetter(string s) {

}
}
```

C Solution:

```
/*
* Problem: Greatest English Letter in Upper and Lower Case
* Difficulty: Easy
* 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* greatestLetter(char* s) {

}
```

Go Solution:

```
// Problem: Greatest English Letter in Upper and Lower Case
// Difficulty: Easy
// 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 greatestLetter(s string) string {

}
```

Kotlin Solution:

```
class Solution {
fun greatestLetter(s: String): String {

}
}
```

Swift Solution:

```
class Solution {
func greatestLetter(_ s: String) -> String {

}
}
```

Rust Solution:

```
// Problem: Greatest English Letter in Upper and Lower Case
// Difficulty: Easy
// 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 greatest_letter(s: String) -> String {

}
}
```

Ruby Solution:

```
# @param {String} s
# @return {String}
def greatest_letter(s)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @return String
*/
function greatestLetter($s) {

}
}
```

Dart Solution:

```
class Solution {
String greatestLetter(String s) {

}
}
```

Scala Solution:

```
object Solution {
def greatestLetter(s: String): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec greatest_letter(s :: String.t) :: String.t
def greatest_letter(s) do

end
end
```

Erlang Solution:

```
-spec greatest_letter(S :: unicode:unicode_binary()) ->
unicode:unicode_binary().
greatest_letter(S) ->
.
```

Racket Solution:

```
(define/contract (greatest-letter s)
(-> string? string?)
)
```