

Problem 784: Letter Case Permutation

Problem Information

Difficulty: Medium

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

Given a string

s

, you can transform every letter individually to be lowercase or uppercase to create another string.

Return

a list of all possible strings we could create

. Return the output in

any order

Example 1:

Input:

s = "a1b2"

Output:

["a1b2", "a1B2", "A1b2", "A1B2"]

Example 2:

Input:

s = "3z4"

Output:

["3z4", "3Z4"]

Constraints:

1 <= s.length <= 12

s

consists of lowercase English letters, uppercase English letters, and digits.

Code Snippets

C++:

```
class Solution {
public:
vector<string> letterCasePermutation(string s) {
    }
};
```

Java:

```
class Solution {
public List<String> letterCasePermutation(String s) {
    }
}
```

Python3:

```
class Solution:  
    def letterCasePermutation(self, s: str) -> List[str]:
```

Python:

```
class Solution(object):  
    def letterCasePermutation(self, s):  
        """  
        :type s: str  
        :rtype: List[str]  
        """
```

JavaScript:

```
/**  
 * @param {string} s  
 * @return {string[]}  
 */  
var letterCasePermutation = function(s) {  
  
};
```

TypeScript:

```
function letterCasePermutation(s: string): string[] {  
  
};
```

C#:

```
public class Solution {  
    public IList<string> LetterCasePermutation(string s) {  
  
    }  
}
```

C:

```
/**  
 * Note: The returned array must be malloced, assume caller calls free().  
 */  
char** letterCasePermutation(char* s, int* returnSize) {
```

```
}
```

Go:

```
func letterCasePermutation(s string) []string {  
}  
}
```

Kotlin:

```
class Solution {  
    fun letterCasePermutation(s: String): List<String> {  
        }  
    }  
}
```

Swift:

```
class Solution {  
    func letterCasePermutation(_ s: String) -> [String] {  
        }  
    }  
}
```

Rust:

```
impl Solution {  
    pub fn letter_case_permutation(s: String) -> Vec<String> {  
        }  
    }  
}
```

Ruby:

```
# @param {String} s  
# @return {String[]}  
def letter_case_permutation(s)  
  
end
```

PHP:

```
class Solution {  
  
    /**  
     * @param String $s  
     * @return String[]  
     */  
    function letterCasePermutation($s) {  
  
    }  
}
```

Dart:

```
class Solution {  
List<String> letterCasePermutation(String s) {  
  
}  
}
```

Scala:

```
object Solution {  
def letterCasePermutation(s: String): List[String] = {  
  
}  
}
```

Elixir:

```
defmodule Solution do  
@spec letter_case_permutation(s :: String.t) :: [String.t]  
def letter_case_permutation(s) do  
  
end  
end
```

Erlang:

```
-spec letter_case_permutation(S :: unicode:unicode_binary()) ->  
[unicode:unicode_binary()].  
letter_case_permutation(S) ->  
.
```

Racket:

```
(define/contract (letter-case-permutation s)
  (-> string? (listof string?))
)
```

Solutions

C++ Solution:

```
/*
 * Problem: Letter Case Permutation
 * Difficulty: Medium
 * 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:
vector<string> letterCasePermutation(string s) {

}

};
```

Java Solution:

```
/**
 * Problem: Letter Case Permutation
 * Difficulty: Medium
 * 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 List<String> letterCasePermutation(String s) {
```

```
}
```

```
}
```

Python3 Solution:

```
"""
Problem: Letter Case Permutation
Difficulty: Medium
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 letterCasePermutation(self, s: str) -> List[str]:
        # TODO: Implement optimized solution
        pass
```

Python Solution:

```
class Solution(object):

    def letterCasePermutation(self, s):
        """
:type s: str
:rtype: List[str]
"""


```

JavaScript Solution:

```
/**
 * Problem: Letter Case Permutation
 * Difficulty: Medium
 * 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} s
 * @return {string[]}
 */
var letterCasePermutation = function(s) {

};

```

TypeScript Solution:

```

/**
 * Problem: Letter Case Permutation
 * Difficulty: Medium
 * 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 letterCasePermutation(s: string): string[] {
}

```

C# Solution:

```

/*
 * Problem: Letter Case Permutation
 * Difficulty: Medium
 * 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 IList<string> LetterCasePermutation(string s) {
    }
}
```

```
}
```

C Solution:

```
/*
 * Problem: Letter Case Permutation
 * Difficulty: Medium
 * 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
 */

/**
 * Note: The returned array must be malloced, assume caller calls free().
 */
char** letterCasePermutation(char* s, int* returnSize) {

}
```

Go Solution:

```
// Problem: Letter Case Permutation
// Difficulty: Medium
// 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 letterCasePermutation(s string) []string {

}
```

Kotlin Solution:

```
class Solution {
    fun letterCasePermutation(s: String): List<String> {
}
```

}

Swift Solution:

```
class Solution {
    func letterCasePermutation(_ s: String) -> [String] {
        }
    }
}
```

Rust Solution:

```
// Problem: Letter Case Permutation
// Difficulty: Medium
// 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 letter_case_permutation(s: String) -> Vec<String> {
        let mut result = vec![String::new()];
        let mut current_index = 0;
        let mut current_string = s;

        while current_index < current_string.len() {
            if current_string[current_index].is_alphabetic() {
                let mut new_result = result.clone();
                let mut new_current_string = current_string.clone();

                new_current_string.replace_range(
                    current_index..=current_index,
                    if new_current_string[current_index].is_uppercase() {
                        new_current_string[current_index].to_lowercase()
                    } else {
                        new_current_string[current_index].to_uppercase()
                    }
                );

                new_result.push(new_current_string);
                result = new_result;
            }

            current_index += 1;
            current_string = result.last().unwrap().clone();
        }

        result
    }
}
```

Ruby Solution:

```
# @param {String} s
# @return {String[]}
def letter_case_permutation(s)

end
```

PHP Solution:

```
class Solution {  
  
    /**  
     * @param String $s  
     * @return String[]  
    */  
    public function lengthOfLongestSubstring($s) {  
        $start = 0;  
        $end = 0;  
        $maxLength = 0;  
        $charIndexMap = array();  
  
        while ($end < strlen($s)) {  
            if (!array_key_exists($s[$end], $charIndexMap)) {  
                $charIndexMap[$s[$end]] = $end;  
                $end++;  
            } else {  
                $start = max($start, $charIndexMap[$s[$end]] + 1);  
                $charIndexMap[$s[$end]] = $end;  
                $end++;  
            }  
            $maxLength = max($maxLength, $end - $start);  
        }  
        return $maxLength;  
    }  
}
```

```
 */
function letterCasePermutation($s) {

}
}
```

Dart Solution:

```
class Solution {
List<String> letterCasePermutation(String s) {

}
}
```

Scala Solution:

```
object Solution {
def letterCasePermutation(s: String): List[String] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec letter_case_permutation(s :: String.t) :: [String.t]
def letter_case_permutation(s) do

end
end
```

Erlang Solution:

```
-spec letter_case_permutation(S :: unicode:unicode_binary()) ->
[unicode:unicode_binary()].
letter_case_permutation(S) ->
.
```

Racket Solution:

```
(define/contract (letter-case-permutation s)
  (-> string? (listof string?))
  )
```