# Problem 917: Reverse Only Letters

## Problem Information

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

## Problem Description

Given a string

s

, reverse the string according to the following rules:

All the characters that are not English letters remain in the same position.

All the English letters (lowercase or uppercase) should be reversed.

Return

s

after reversing it

.

Example 1:

Input:

s = "ab-cd"

Output:

"dc-ba"

Example 2:

Input:

s = "a-bC-dEf-ghIj"

Output:

"j-Ih-gfE-dCba"

Example 3:

Input:

s = "Test1ng-Leet=code-Q!"

Output:

"Qedo1ct-eeLg=ntse-T!"

Constraints:

1 <= s.length <= 100

s

consists of characters with ASCII values in the range

[33, 122]

.

s

does not contain

'\"'

or

'\\'

.

## Code Snippets

### C++:

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

}
};
```

### Java:

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

}
}
```

### Python3:

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

### Python:

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

### JavaScript:

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

};
```

**TypeScript:**

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

};
```

**C#:**

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

}
}
```

**C:**

```
char* reverseOnlyLetters(char* s) {

}
```

**Go:**

```
func reverseOnlyLetters(s string) string {

}
```

**Kotlin:**

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

}
}
```

**Swift:**

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

}
}
```

**Rust:**

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

}
}
```

**Ruby:**

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

end
```

**PHP:**

```
class Solution {

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

}
}
```

**Dart:**

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

}
}
```

### Scala:

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

}
}
```

### Elixir:

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

end
end
```

### Erlang:

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

### Racket:

```
(define/contract (reverse-only-letters s)
(-> string? string?)
)
```

## Solutions

### C++ Solution:

```
/*
* Problem: Reverse Only Letters
* Difficulty: Easy
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
```

```
*/

class Solution {
public:
string reverseOnlyLetters(string s) {

}
};
```

### Java Solution:

```
/**
* Problem: Reverse Only Letters
* Difficulty: Easy
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public String reverseOnlyLetters(String s) {

}
}
```

### Python3 Solution:

```
"""
Problem: Reverse Only Letters
Difficulty: Easy
Tags: array, string

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

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

```
# TODO: Implement optimized solution
pass
```

### Python Solution:

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

### JavaScript Solution:

```
/**
* Problem: Reverse Only Letters
* Difficulty: Easy
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* 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 reverseOnlyLetters = function(s) {

};
```

### TypeScript Solution:

```
/**
* Problem: Reverse Only Letters
* Difficulty: Easy
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function reverseOnlyLetters(s: string): string {

};
```

### C# Solution:

```
/*
* Problem: Reverse Only Letters
* Difficulty: Easy
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public string ReverseOnlyLetters(string s) {

}
}
```

### C Solution:

```
/*
* Problem: Reverse Only Letters
* Difficulty: Easy
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

char* reverseOnlyLetters(char* s) {

}
```

### Go Solution:

```
// Problem: Reverse Only Letters
// Difficulty: Easy
// Tags: array, string
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func reverseOnlyLetters(s string) string {

}
```

### Kotlin Solution:

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

}
}
```

### Swift Solution:

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

}
}
```

### Rust Solution:

```
// Problem: Reverse Only Letters
// Difficulty: Easy
// Tags: array, string
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

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

**Ruby Solution:**

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

end
```

**PHP Solution:**

```
class Solution {

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

}
}
```

**Dart Solution:**

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

}
}
```

**Scala Solution:**

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

}
}
```

## Elixir Solution:

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

end
end
```

## Erlang Solution:

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

## Racket Solution:

```
(define/contract (reverse-only-letters s)
(-> string? string?)
)
```