Problem 2788: Split Strings by Separator

Problem Information

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

Problem Description

Given an array of strings

words

and a character

separator

,

split

each string in

words

by

separator

.

Return

an array of strings containing the new strings formed after the splits,

excluding empty strings

.

Notes

separator

is used to determine where the split should occur, but it is not included as part of the resulting strings.

A split may result in more than two strings.

The resulting strings must maintain the same order as they were initially given.

Example 1:

Input:

words = ["one.two.three","four.five","six"], separator = "."

Output:

["one","two","three","four","five","six"]

Explanation:

In this example we split as follows:

"one.two.three" splits into "one", "two", "three" "four.five" splits into "four", "five" "six" splits into "six"

Hence, the resulting array is ["one","two","three","four","five","six"].

Example 2:

Input:

words = ["$easy$","$problem$"], separator = "$"

Output:

["easy","problem"]

Explanation:

In this example we split as follows:

"$easy$" splits into "easy" (excluding empty strings) "$problem$" splits into "problem" (excluding empty strings)

Hence, the resulting array is ["easy","problem"].

Example 3:

Input:

words = ["|||"], separator = "|"

Output:

[]

Explanation:

In this example the resulting split of "|||" will contain only empty strings, so we return an empty array [].

Constraints:

1 <= words.length <= 100

1 <= words[i].length <= 20

characters in

words[i]

are either lowercase English letters or characters from the string

".,|$#@"

(excluding the quotes)

separator

is a character from the string

".,|$#@"

(excluding the quotes)

Code Snippets

C++:

```
class Solution {
public:
vector<string> splitWordsBySeparator(vector<string>& words, char separator) {

}
};
```

Java:

```
class Solution {
public List<String> splitWordsBySeparator(List<String> words, char separator)
{

}
}
```

Python3:

```
class Solution:
def splitWordsBySeparator(self, words: List[str], separator: str) ->
List[str]:
```

Python:

```
class Solution(object):
def splitWordsBySeparator(self, words, separator):
"""
:type words: List[str]
:type separator: str
:rtype: List[str]
"""
```

JavaScript:

```
/**
* @param {string[]} words
* @param {character} separator
* @return {string[]}
*/
var splitWordsBySeparator = function(words, separator) {

};
```

TypeScript:

```
function splitWordsBySeparator(words: string[], separator: string): string[]
{

};
```

C#:

```
public class Solution {
public IList<string> SplitWordsBySeparator(IList<string> words, char
separator) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
char** splitWordsBySeparator(char** words, int wordsSize, char separator,
int* returnSize) {
```

```
}
```

Go:

```
func splitWordsBySeparator(words []string, separator byte) []string {

}
```

Kotlin:

```
class Solution {
fun splitWordsBySeparator(words: List<String>, separator: Char): List<String>
{

}
}
```

Swift:

```
class Solution {
func splitWordsBySeparator(_ words: [String], _ separator: Character) ->
[String] {

}
}
```

Rust:

```
impl Solution {
pub fn split_words_by_separator(words: Vec<String>, separator: char) ->
Vec<String> {

}
}
```

Ruby:

```
# @param {String[]} words
# @param {Character} separator
# @return {String[]}
def split_words_by_separator(words, separator)
```

```
end
```

PHP:

```
class Solution {

/**
* @param String[] $words
* @param String $separator
* @return String[]
*/
function splitWordsBySeparator($words, $separator) {

}
}
```

Dart:

```
class Solution {
List<String> splitWordsBySeparator(List<String> words, String separator) {

}
}
```

Scala:

```
object Solution {
def splitWordsBySeparator(words: List[String], separator: Char): List[String]
= {

}
}
```

Elixir:

```
defmodule Solution do
@spec split_words_by_separator(words :: [String.t], separator :: char) ::
[String.t]
def split_words_by_separator(words, separator) do

end
end
```

Erlang:

```
-spec split_words_by_separator(Words :: [unicode:unicode_binary()], Separator
:: char()) -> [unicode:unicode_binary()].
split_words_by_separator(Words, Separator) ->
.
```

Racket:

```
(define/contract (split-words-by-separator words separator)
(-> (listof string?) char? (listof string?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Split Strings by Separator
* 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:
vector<string> splitWordsBySeparator(vector<string>& words, char separator) {

}
};
```

Java Solution:

```
/**
* Problem: Split Strings by Separator
* 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 List<String> splitWordsBySeparator(List<String> words, char separator)
{

}
}
```

Python3 Solution:

```
"""
Problem: Split Strings by Separator
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 splitWordsBySeparator(self, words: List[str], separator: str) ->
List[str]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def splitWordsBySeparator(self, words, separator):
"""
:type words: List[str]
:type separator: str
:rtype: List[str]
"""
```

JavaScript Solution:

```
/**
* Problem: Split Strings by Separator
* 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[]} words
* @param {character} separator
* @return {string[]}
*/
var splitWordsBySeparator = function(words, separator) {

};
```

TypeScript Solution:

```
/**
* Problem: Split Strings by Separator
* 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 splitWordsBySeparator(words: string[], separator: string): string[]
{

};
```

C# Solution:

```
/*
* Problem: Split Strings by Separator
* 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 IList<string> SplitWordsBySeparator(IList<string> words, char
separator) {

}
}
```

C Solution:

```
/*
* Problem: Split Strings by Separator
* 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
*/

/**
* Note: The returned array must be malloced, assume caller calls free().
*/
char** splitWordsBySeparator(char** words, int wordsSize, char separator,
int* returnSize) {

}
```

Go Solution:

```
// Problem: Split Strings by Separator
// 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 splitWordsBySeparator(words []string, separator byte) []string {

}
```

Kotlin Solution:

```
class Solution {
fun splitWordsBySeparator(words: List<String>, separator: Char): List<String>
{

}
}
```

Swift Solution:

```
class Solution {
func splitWordsBySeparator(_ words: [String], _ separator: Character) ->
[String] {

}
}
```

Rust Solution:

```
// Problem: Split Strings by Separator
// 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 split_words_by_separator(words: Vec<String>, separator: char) ->
Vec<String> {

}
}
```

Ruby Solution:

```
# @param {String[]} words
# @param {Character} separator
# @return {String[]}
def split_words_by_separator(words, separator)

end
```

PHP Solution:

```
class Solution {

/**
* @param String[] $words
* @param String $separator
* @return String[]
*/
function splitWordsBySeparator($words, $separator) {

}
}
```

Dart Solution:

```
class Solution {
List<String> splitWordsBySeparator(List<String> words, String separator) {

}
}
```

Scala Solution:

```
object Solution {
def splitWordsBySeparator(words: List[String], separator: Char): List[String]
= {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec split_words_by_separator(words :: [String.t], separator :: char) ::
[String.t]
def split_words_by_separator(words, separator) do

end
end
```

Erlang Solution:

```
-spec split_words_by_separator(Words :: [unicode:unicode_binary()], Separator
:: char()) -> [unicode:unicode_binary()].
split_words_by_separator(Words, Separator) ->
.
```

Racket Solution:

```
(define/contract (split-words-by-separator words separator)
(-> (listof string?) char? (listof string?))
)
```