# Problem 1427: Perform String Shifts

## Problem Information

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

## Problem Description

You are given a string

s

containing lowercase English letters, and a matrix

shift

, where

shift[i] = [direction

i

, amount

i

]

:

direction

i

can be

0

(for left shift) or

1

(for right shift).

amount

i

is the amount by which string

s

is to be shifted.

A left shift by 1 means remove the first character of

s

and append it to the end.

Similarly, a right shift by 1 means remove the last character of

s

and add it to the beginning.

Return the final string after all operations.

Example 1:

Input:

s = "abc", shift = [[0,1],[1,2]]

Output:

"cab"

Explanation:

[0,1] means shift to left by 1. "abc" -> "bca" [1,2] means shift to right by 2. "bca" -> "cab"

Example 2:

Input:

s = "abcdefg", shift = [[1,1],[1,1],[0,2],[1,3]]

Output:

"efgabcd"

Explanation:

[1,1] means shift to right by 1. "abcdefg" -> "gabcdef" [1,1] means shift to right by 1. "gabcdef" -> "fgabcde" [0,2] means shift to left by 2. "fgabcde" -> "abcdefg" [1,3] means shift to right by 3. "abcdefg" -> "efgabcd"

Constraints:

1 <= s.length <= 100

s

only contains lower case English letters.

1 <= shift.length <= 100

shift[i].length == 2

direction

i

is either

0

or

1

.

0 <= amount

i

<= 100

## Code Snippets

**C++:**

```
class Solution {
public:
string stringShift(string s, vector<vector<int>>& shift) {

}
};
```

**Java:**

```
class Solution {
public String stringShift(String s, int[][] shift) {

}
}
```

**Python3:**

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

**Python:**

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

**JavaScript:**

```
/**
* @param {string} s
* @param {number[][]} shift
* @return {string}
*/
var stringShift = function(s, shift) {

};
```

**TypeScript:**

```
function stringShift(s: string, shift: number[][]): string {

};
```

**C#:**

```
public class Solution {
public string StringShift(string s, int[][] shift) {

}
}
```

**C:**

```
char* stringShift(char* s, int** shift, int shiftSize, int* shiftColSize) {

}
```

**Go:**

```
func stringShift(s string, shift [][]int) string {

}
```

**Kotlin:**

```
class Solution {
fun stringShift(s: String, shift: Array<IntArray>): String {

}
}
```

**Swift:**

```
class Solution {
func stringShift(_ s: String, _ shift: [[Int]]) -> String {

}
}
```

**Rust:**

```
impl Solution {
pub fn string_shift(s: String, shift: Vec<Vec<i32>>) -> String {

}
}
```

**Ruby:**

```
# @param {String} s
# @param {Integer[][]} shift
# @return {String}
def string_shift(s, shift)

end
```

**PHP:**

```
class Solution {

/**
* @param String $s
```

```
* @param Integer[][] $shift
* @return String
*/
function stringShift($s, $shift) {

}
}
```

**Dart:**

```
class Solution {
String stringShift(String s, List<List<int>> shift) {

}
}
```

**Scala:**

```
object Solution {
def stringShift(s: String, shift: Array[Array[Int]]): String = {

}
}
```

**Elixir:**

```
defmodule Solution do
@spec string_shift(s :: String.t, shift :: [[integer]]) :: String.t
def string_shift(s, shift) do

end
end
```

**Erlang:**

```
-spec string_shift(S :: unicode:unicode_binary(), Shift :: [[integer()]]) ->
unicode:unicode_binary().
string_shift(S, Shift) ->
.
```

**Racket:**

```
(define/contract (string-shift s shift)
(-> string? (listof (listof exact-integer?)) string?)
)
```

## Solutions

### C++ Solution:

```
/*
* Problem: Perform String Shifts
* Difficulty: Easy
* Tags: array, string, math
*
* 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 stringShift(string s, vector<vector<int>>& shift) {

}
};
```

### Java Solution:

```
/**
* Problem: Perform String Shifts
* Difficulty: Easy
* Tags: array, string, math
*
* 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 stringShift(String s, int[][] shift) {

}
```

```
}
```

### Python3 Solution:

```
"""
Problem: Perform String Shifts
Difficulty: Easy
Tags: array, string, math

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 stringShift(self, s: str, shift: List[List[int]]) -> str:
# TODO: Implement optimized solution
pass
```

### Python Solution:

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

### JavaScript Solution:

```
/**
* Problem: Perform String Shifts
* Difficulty: Easy
* Tags: array, string, math
*
* 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
* @param {number[][]} shift
* @return {string}
*/
var stringShift = function(s, shift) {

};
```

### TypeScript Solution:

```
/**
* Problem: Perform String Shifts
* Difficulty: Easy
* Tags: array, string, math
*
* 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 stringShift(s: string, shift: number[][]): string {

};
```

### C# Solution:

```
/*
* Problem: Perform String Shifts
* Difficulty: Easy
* Tags: array, string, math
*
* 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 StringShift(string s, int[][] shift) {

}
```

```
}
```

### C Solution:

```
/*
* Problem: Perform String Shifts
* Difficulty: Easy
* Tags: array, string, math
*
* 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* stringShift(char* s, int** shift, int shiftSize, int* shiftColSize) {

}
```

### Go Solution:

```
// Problem: Perform String Shifts
// Difficulty: Easy
// Tags: array, string, math
//
// 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 stringShift(s string, shift [][]int) string {

}
```

### Kotlin Solution:

```
class Solution {
fun stringShift(s: String, shift: Array<IntArray>): String {

}
}
```

### Swift Solution:

```
class Solution {
func stringShift(_ s: String, _ shift: [[Int]]) -> String {

}
}
```

### Rust Solution:

```
// Problem: Perform String Shifts
// Difficulty: Easy
// Tags: array, string, math
//
// 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 string_shift(s: String, shift: Vec<Vec<i32>>) -> String {

}
}
```

### Ruby Solution:

```
# @param {String} s
# @param {Integer[][]} shift
# @return {String}
def string_shift(s, shift)

end
```

### PHP Solution:

```
class Solution {

/**
* @param String $s
* @param Integer[][] $shift
* @return String
*/
function stringShift($s, $shift) {
```

```
}
}
```

**Dart Solution:**

```
class Solution {
String stringShift(String s, List<List<int>> shift) {

}
}
```

**Scala Solution:**

```
object Solution {
def stringShift(s: String, shift: Array[Array[Int]]): String = {

}
}
```

**Elixir Solution:**

```
defmodule Solution do
@spec string_shift(s :: String.t, shift :: [[integer]]) :: String.t
def string_shift(s, shift) do

end
end
```

**Erlang Solution:**

```
-spec string_shift(S :: unicode:unicode_binary(), Shift :: [[integer()]]) ->
unicode:unicode_binary().
string_shift(S, Shift) ->
.
```

**Racket Solution:**

```
(define/contract (string-shift s shift)
(-> string? (listof (listof exact-integer?)) string?)
)
```