Problem 2288: Apply Discount to Prices

Problem Information

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

Problem Description

A

sentence

is a string of single-space separated words where each word can contain digits, lowercase letters, and the dollar sign

'$'

. A word represents a

price

if it is a sequence of digits preceded by a dollar sign.

For example,

"$100"

,

"$23"

, and

"$6"

represent prices while

"100"

,

"$"

, and

"$1e5"

do not.

You are given a string

sentence

representing a sentence and an integer

discount

. For each word representing a price, apply a discount of

discount%

on the price and

update

the word in the sentence. All updated prices should be represented with

exactly two

decimal places.

Return

a string representing the modified sentence

.

Note that all prices will contain

at most

10

digits.

Example 1:

Input:

sentence = "there are $1 $2 and 5$ candies in the shop", discount = 50

Output:

"there are $0.50 $1.00 and 5$ candies in the shop"

Explanation:

The words which represent prices are "$1" and "$2". - A 50% discount on "$1" yields "$0.50", so "$1" is replaced by "$0.50". - A 50% discount on "$2" yields "$1". Since we need to have exactly 2 decimal places after a price, we replace "$2" with "$1.00".

Example 2:

Input:

sentence = "1 2 $3 4 $5 $6 7 8$ $9 10", discount = 100

Output:

"1 2 $0.00 4 $0.00 $0.00 7 8$ $0.00 10"

Explanation:

Applying a 100% discount on any price will result in 0. The words representing prices are "$3", "$5", "$6", and "$9". Each of them is replaced by "$0.00".

Constraints:

1 <= sentence.length <= 10

5

sentence

consists of lowercase English letters, digits,

' '

, and

'$'

.

sentence

does not have leading or trailing spaces.

All words in

sentence

are separated by a single space.

All prices will be

positive

numbers without leading zeros.

All prices will have

at most

10

digits.

0 <= discount <= 100

Code Snippets

C++:

```
class Solution {
public:
string discountPrices(string sentence, int discount) {

}
};
```

Java:

```
class Solution {
public String discountPrices(String sentence, int discount) {

}
}
```

Python3:

```
class Solution:
def discountPrices(self, sentence: str, discount: int) -> str:
```

Python:

```
class Solution(object):
def discountPrices(self, sentence, discount):
"""
:type sentence: str
:type discount: int
:rtype: str
```

```
"""
```

JavaScript:

```
/**
* @param {string} sentence
* @param {number} discount
* @return {string}
*/
var discountPrices = function(sentence, discount) {

};
```

TypeScript:

```
function discountPrices(sentence: string, discount: number): string {

};
```

C#:

```
public class Solution {
public string DiscountPrices(string sentence, int discount) {

}
}
```

C:

```
char* discountPrices(char* sentence, int discount) {

}
```

Go:

```
func discountPrices(sentence string, discount int) string {

}
```

Kotlin:

```
class Solution {
fun discountPrices(sentence: String, discount: Int): String {

}
}
```

Swift:

```
class Solution {
func discountPrices(_ sentence: String, _ discount: Int) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn discount_prices(sentence: String, discount: i32) -> String {

}
}
```

Ruby:

```
# @param {String} sentence
# @param {Integer} discount
# @return {String}
def discount_prices(sentence, discount)

end
```

PHP:

```
class Solution {

/**
* @param String $sentence
* @param Integer $discount
* @return String
*/
function discountPrices($sentence, $discount) {

}
```

```
}
```

Dart:

```
class Solution {
String discountPrices(String sentence, int discount) {

}
}
```

Scala:

```
object Solution {
def discountPrices(sentence: String, discount: Int): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec discount_prices(sentence :: String.t, discount :: integer) :: String.t
def discount_prices(sentence, discount) do

end
end
```

Erlang:

```
-spec discount_prices(Sentence :: unicode:unicode_binary(), Discount ::
integer()) -> unicode:unicode_binary().
discount_prices(Sentence, Discount) ->
.
```

Racket:

```
(define/contract (discount-prices sentence discount)
(-> string? exact-integer? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Apply Discount to Prices
* 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:
string discountPrices(string sentence, int discount) {

}
};
```

Java Solution:

```
/**
* Problem: Apply Discount to Prices
* 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 String discountPrices(String sentence, int discount) {

}
}
```

Python3 Solution:

```
"""
Problem: Apply Discount to Prices
```

```
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 discountPrices(self, sentence: str, discount: int) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def discountPrices(self, sentence, discount):
"""
:type sentence: str
:type discount: int
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Apply Discount to Prices
* 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} sentence
* @param {number} discount
* @return {string}
*/
var discountPrices = function(sentence, discount) {
```

```
};
```

TypeScript Solution:

```
/**
* Problem: Apply Discount to Prices
* 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 discountPrices(sentence: string, discount: number): string {

};
```

C# Solution:

```
/*
* Problem: Apply Discount to Prices
* 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 string DiscountPrices(string sentence, int discount) {

}
}
```

C Solution:

```
/*
* Problem: Apply Discount to Prices
```

```
* 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
*/

char* discountPrices(char* sentence, int discount) {

}
```

Go Solution:

```
// Problem: Apply Discount to Prices
// 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 discountPrices(sentence string, discount int) string {

}
```

Kotlin Solution:

```
class Solution {
fun discountPrices(sentence: String, discount: Int): String {

}
}
```

Swift Solution:

```
class Solution {
func discountPrices(_ sentence: String, _ discount: Int) -> String {

}
}
```

Rust Solution:

```
// Problem: Apply Discount to Prices
// 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 discount_prices(sentence: String, discount: i32) -> String {

}
}
```

Ruby Solution:

```
# @param {String} sentence
# @param {Integer} discount
# @return {String}
def discount_prices(sentence, discount)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $sentence
* @param Integer $discount
* @return String
*/
function discountPrices($sentence, $discount) {

}
}
```

Dart Solution:

```
class Solution {
String discountPrices(String sentence, int discount) {

}
}
```

Scala Solution:

```
object Solution {
def discountPrices(sentence: String, discount: Int): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec discount_prices(sentence :: String.t, discount :: integer) :: String.t
def discount_prices(sentence, discount) do

end
end
```

Erlang Solution:

```
-spec discount_prices(Sentence :: unicode:unicode_binary(), Discount ::
integer()) -> unicode:unicode_binary().
discount_prices(Sentence, Discount) ->
.
```

Racket Solution:

```
(define/contract (discount-prices sentence discount)
(-> string? exact-integer? string?)
)
```