Problem 1556: Thousand Separator

Problem Information

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

Problem Description

Given an integer

n

, add a dot (".") as the thousands separator and return it in string format.

Example 1:

Input:

n = 987

Output:

"987"

Example 2:

Input:

n = 1234

Output:

"1.234"

Constraints:

0 <= n <= 2

31

- 1

Code Snippets

C++:

```
class Solution {
public:
string thousandSeparator(int n) {

}
};
```

Java:

```
class Solution {
public String thousandSeparator(int n) {

}
}
```

Python3:

```
class Solution:
def thousandSeparator(self, n: int) -> str:
```

Python:

```
class Solution(object):
def thousandSeparator(self, n):
"""
:type n: int
:rtype: str
"""
```

JavaScript:

```
/**
* @param {number} n
* @return {string}
*/
var thousandSeparator = function(n) {

};
```

TypeScript:

```
function thousandSeparator(n: number): string {

};
```

C#:

```
public class Solution {
public string ThousandSeparator(int n) {

}
}
```

C:

```
char* thousandSeparator(int n) {

}
```

Go:

```
func thousandSeparator(n int) string {

}
```

Kotlin:

```
class Solution {
fun thousandSeparator(n: Int): String {

}
}
```

Swift:

```
class Solution {
func thousandSeparator(_ n: Int) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn thousand_separator(n: i32) -> String {

}
}
```

Ruby:

```
# @param {Integer} n
# @return {String}
def thousand_separator(n)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @return String
*/
function thousandSeparator($n) {

}
}
```

Dart:

```
class Solution {
String thousandSeparator(int n) {

}
```

```
}
```

Scala:

```
object Solution {
def thousandSeparator(n: Int): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec thousand_separator(n :: integer) :: String.t
def thousand_separator(n) do

end
end
```

Erlang:

```
-spec thousand_separator(N :: integer()) -> unicode:unicode_binary().
thousand_separator(N) ->
.
```

Racket:

```
(define/contract (thousand-separator n)
(-> exact-integer? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Thousand Separator
* Difficulty: Easy
* 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 thousandSeparator(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Thousand Separator
* Difficulty: Easy
* 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 thousandSeparator(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Thousand Separator
Difficulty: Easy
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 thousandSeparator(self, n: int) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def thousandSeparator(self, n):
"""
:type n: int
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Thousand Separator
* Difficulty: Easy
* 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 {number} n
* @return {string}
*/
var thousandSeparator = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: Thousand Separator
* Difficulty: Easy
* 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 thousandSeparator(n: number): string {

};
```

C# Solution:

```
/*
* Problem: Thousand Separator
* Difficulty: Easy
* 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 ThousandSeparator(int n) {

}
}
```

C Solution:

```
/*
* Problem: Thousand Separator
* Difficulty: Easy
* 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* thousandSeparator(int n) {
```

```
}
```

Go Solution:

```
// Problem: Thousand Separator
// Difficulty: Easy
// 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 thousandSeparator(n int) string {

}
```

Kotlin Solution:

```
class Solution {
fun thousandSeparator(n: Int): String {

}
}
```

Swift Solution:

```
class Solution {
func thousandSeparator(_ n: Int) -> String {

}
}
```

Rust Solution:

```
// Problem: Thousand Separator
// Difficulty: Easy
// 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 thousand_separator(n: i32) -> String {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @return {String}
def thousand_separator(n)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @return String
*/
function thousandSeparator($n) {

}
}
```

Dart Solution:

```
class Solution {
String thousandSeparator(int n) {

}
}
```

Scala Solution:

```
object Solution {
def thousandSeparator(n: Int): String = {
```

```
}
}
```

Elixir Solution:

```
defmodule Solution do
@spec thousand_separator(n :: integer) :: String.t
def thousand_separator(n) do

end
end
```

Erlang Solution:

```
-spec thousand_separator(N :: integer()) -> unicode:unicode_binary().
thousand_separator(N) ->
.
```

Racket Solution:

```
(define/contract (thousand-separator n)
(-> exact-integer? string?)
)
```