Problem 12: Integer to Roman

Problem Information

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

Problem Description

Seven different symbols represent Roman numerals with the following values:

Symbol

Value

I

1

V

5

X

10

L

50

C

100

D

500

M

1000

Roman numerals are formed by appending the conversions of decimal place values from highest to lowest. Converting a decimal place value into a Roman numeral has the following rules:

If the value does not start with 4 or 9, select the symbol of the maximal value that can be subtracted from the input, append that symbol to the result, subtract its value, and convert the remainder to a Roman numeral.

If the value starts with 4 or 9 use the

subtractive form

representing one symbol subtracted from the following symbol, for example, 4 is 1 (

I

) less than 5 (

V

):

IV

and 9 is 1 (

I

) less than 10 (

X

):

IX

. Only the following subtractive forms are used: 4 (

IV

), 9 (

IX

), 40 (

XL

), 90 (

XC

), 400 (

CD

) and 900 (

CM

).

Only powers of 10 (

I

,

X

,

C

,

M

) can be appended consecutively at most 3 times to represent multiples of 10. You cannot append 5 (

V

), 50 (

L

), or 500 (

D

) multiple times. If you need to append a symbol 4 times use the

subtractive form

.

Given an integer, convert it to a Roman numeral.

Example 1:

Input:

num = 3749

Output:

"MMMDCCXLIX"

Explanation:

3000 = MMM as 1000 (M) + 1000 (M) + 1000 (M) 700 = DCC as 500 (D) + 100 (C) + 100 (C) 40 = XL as 10 (X) less of 50 (L) 9 = IX as 1 (I) less of 10 (X) Note: 49 is not 1 (I) less of 50 (L) because the conversion is based on decimal places

Example 2:

Input:

num = 58

Output:

"LVIII"

Explanation:

50 = L 8 = VIII

Example 3:

Input:

num = 1994

Output:

"MCMXCIV"

Explanation:

1000 = M 900 = CM 90 = XC 4 = IV

Constraints:

1 <= num <= 3999

Code Snippets

C++:

```
class Solution {
public:
string intToRoman(int num) {

}
};
```

Java:

```
class Solution {
public String intToRoman(int num) {

}
}
```

Python3:

```
class Solution:
def intToRoman(self, num: int) -> str:
```

Python:

```
class Solution(object):
def intToRoman(self, num):
"""
:type num: int
:rtype: str
"""
```

JavaScript:

```
/**
* @param {number} num
* @return {string}
*/
var intToRoman = function(num) {

};
```

TypeScript:

```
function intToRoman(num: number): string {

};
```

C#:

```
public class Solution {
public string IntToRoman(int num) {

}
}
```

C:

```
char* intToRoman(int num) {

}
```

Go:

```
func intToRoman(num int) string {

}
```

Kotlin:

```
class Solution {
fun intToRoman(num: Int): String {

}
}
```

Swift:

```
class Solution {
func intToRoman(_ num: Int) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn int_to_roman(num: i32) -> String {

}
}
```

Ruby:

```
# @param {Integer} num
# @return {String}
def int_to_roman(num)

end
```

PHP:

```
class Solution {

/**
* @param Integer $num
* @return String
*/
function intToRoman($num) {

}
}
```

Dart:

```
class Solution {
String intToRoman(int num) {

}
}
```

Scala:

```
object Solution {
def intToRoman(num: Int): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec int_to_roman(num :: integer) :: String.t
def int_to_roman(num) do

end
end
```

Erlang:

```
-spec int_to_roman(Num :: integer()) -> unicode:unicode_binary().
int_to_roman(Num) ->
.
```

Racket:

```
(define/contract (int-to-roman num)
(-> exact-integer? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Integer to Roman
* Difficulty: Medium
* Tags: string, math, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
string intToRoman(int num) {

}
};
```

Java Solution:

```
/**
* Problem: Integer to Roman
* Difficulty: Medium
* Tags: string, math, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public String intToRoman(int num) {

}
}
```

Python3 Solution:

```
"""
Problem: Integer to Roman
Difficulty: Medium
Tags: string, math, hash

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class Solution:
def intToRoman(self, num: int) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def intToRoman(self, num):
"""
:type num: int
:rtype: str
```

```
"""
```

JavaScript Solution:

```
/**
* Problem: Integer to Roman
* Difficulty: Medium
* Tags: string, math, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {number} num
* @return {string}
*/
var intToRoman = function(num) {

};
```

TypeScript Solution:

```
/**
* Problem: Integer to Roman
* Difficulty: Medium
* Tags: string, math, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function intToRoman(num: number): string {

};
```

C# Solution:

```
/*
* Problem: Integer to Roman
* Difficulty: Medium
* Tags: string, math, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public string IntToRoman(int num) {

}
}
```

C Solution:

```
/*
* Problem: Integer to Roman
* Difficulty: Medium
* Tags: string, math, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

char* intToRoman(int num) {

}
```

Go Solution:

```
// Problem: Integer to Roman
// Difficulty: Medium
// Tags: string, math, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map
```

```
func intToRoman(num int) string {

}
```

Kotlin Solution:

```
class Solution {
fun intToRoman(num: Int): String {

}
}
```

Swift Solution:

```
class Solution {
func intToRoman(_ num: Int) -> String {

}
}
```

Rust Solution:

```
// Problem: Integer to Roman
// Difficulty: Medium
// Tags: string, math, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn int_to_roman(num: i32) -> String {

}
}
```

Ruby Solution:

```
# @param {Integer} num
# @return {String}
def int_to_roman(num)
```

```
end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $num
* @return String
*/
function intToRoman($num) {

}
}
```

Dart Solution:

```
class Solution {
String intToRoman(int num) {

}
}
```

Scala Solution:

```
object Solution {
def intToRoman(num: Int): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec int_to_roman(num :: integer) :: String.t
def int_to_roman(num) do

end
end
```

Erlang Solution:

```
-spec int_to_roman(Num :: integer()) -> unicode:unicode_binary().
int_to_roman(Num) ->
.
```

Racket Solution:

```
(define/contract (int-to-roman num)
(-> exact-integer? string?)
)
```