Problem 3602: Hexadecimal and Hexatrigesimal Conversion

Problem Information

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

Problem Description

You are given an integer

n

.

Return the concatenation of the

hexadecimal

representation of

n

2

and the

hexatrigesimal

representation of

n

3

.

A

hexadecimal

number is defined as a base-16 numeral system that uses the digits

0 – 9

and the uppercase letters

A - F

to represent values from 0 to 15.

A

hexatrigesimal

number is defined as a base-36 numeral system that uses the digits

0 – 9

and the uppercase letters

A - Z

to represent values from 0 to 35.

Example 1:

Input:

n = 13

Output:

"A91P1"

Explanation:

n

2

= 13 * 13 = 169

. In hexadecimal, it converts to

(10 * 16) + 9 = 169

, which corresponds to

"A9"

.

n

3

= 13 * 13 * 13 = 2197

. In hexatrigesimal, it converts to

(1 * 36

2

) + (25 * 36) + 1 = 2197

, which corresponds to

"1P1"

.

Concatenating both results gives

"A9" + "1P1" = "A91P1"

.

Example 2:

Input:

n = 36

Output:

"5101000"

Explanation:

n

2

= 36 * 36 = 1296

. In hexadecimal, it converts to

(5 * 16

2

) + (1 * 16) + 0 = 1296

, which corresponds to

"510"

.

n

3

= 36 * 36 * 36 = 46656

. In hexatrigesimal, it converts to

(1 * 36

3

) + (0 * 36

2

) + (0 * 36) + 0 = 46656

, which corresponds to

"1000"

.

Concatenating both results gives

"510" + "1000" = "5101000"

.

Constraints:

1 <= n <= 1000

Code Snippets

C++:

```
class Solution {
public:
string concatHex36(int n) {

}
};
```

Java:

```
class Solution {
public String concatHex36(int n) {

}
}
```

Python3:

```
class Solution:
def concatHex36(self, n: int) -> str:
```

Python:

```
class Solution(object):
def concatHex36(self, n):
"""
:type n: int
:rtype: str
"""
```

JavaScript:

```
/**
* @param {number} n
* @return {string}
*/
var concatHex36 = function(n) {

};
```

TypeScript:

```
function concatHex36(n: number): string {
```

```
};
```

C#:

```
public class Solution {
public string ConcatHex36(int n) {

}
}
```

C:

```
char* concatHex36(int n) {

}
```

Go:

```
func concatHex36(n int) string {

}
```

Kotlin:

```
class Solution {
fun concatHex36(n: Int): String {

}
}
```

Swift:

```
class Solution {
func concatHex36(_ n: Int) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn concat_hex36(n: i32) -> String {
```

```
}
}
```

Ruby:

```
# @param {Integer} n
# @return {String}
def concat_hex36(n)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @return String
*/
function concatHex36($n) {

}
}
```

Dart:

```
class Solution {
String concatHex36(int n) {

}
}
```

Scala:

```
object Solution {
def concatHex36(n: Int): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec concat_hex36(n :: integer) :: String.t
def concat_hex36(n) do

end
end
```

Erlang:

```
-spec concat_hex36(N :: integer()) -> unicode:unicode_binary().
concat_hex36(N) ->
.
```

Racket:

```
(define/contract (concat-hex36 n)
(-> exact-integer? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Hexadecimal and Hexatrigesimal Conversion
* Difficulty: Easy
* Tags: string, math
*
* 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 concatHex36(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Hexadecimal and Hexatrigesimal Conversion
* Difficulty: Easy
* Tags: string, math
*
* 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 concatHex36(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Hexadecimal and Hexatrigesimal Conversion
Difficulty: Easy
Tags: string, math

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 concatHex36(self, n: int) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def concatHex36(self, n):
"""
:type n: int
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Hexadecimal and Hexatrigesimal Conversion
* Difficulty: Easy
* Tags: string, math
*
* 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 concatHex36 = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: Hexadecimal and Hexatrigesimal Conversion
* Difficulty: Easy
* Tags: string, math
*
* 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 concatHex36(n: number): string {

};
```

C# Solution:

```
/*
* Problem: Hexadecimal and Hexatrigesimal Conversion
* Difficulty: Easy
* Tags: string, math
*
```

```
* 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 ConcatHex36(int n) {

}
}
```

C Solution:

```
/*
* Problem: Hexadecimal and Hexatrigesimal Conversion
* Difficulty: Easy
* Tags: string, math
*
* 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* concatHex36(int n) {

}
```

Go Solution:

```
// Problem: Hexadecimal and Hexatrigesimal Conversion
// Difficulty: Easy
// Tags: string, math
//
// 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 concatHex36(n int) string {

}
```

Kotlin Solution:

```
class Solution {
fun concatHex36(n: Int): String {

}
}
```

Swift Solution:

```
class Solution {
func concatHex36(_ n: Int) -> String {

}
}
```

Rust Solution:

```
// Problem: Hexadecimal and Hexatrigesimal Conversion
// Difficulty: Easy
// Tags: string, math
//
// 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 concat_hex36(n: i32) -> String {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @return {String}
def concat_hex36(n)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @return String
*/
function concatHex36($n) {

}
}
```

Dart Solution:

```
class Solution {
String concatHex36(int n) {

}
}
```

Scala Solution:

```
object Solution {
def concatHex36(n: Int): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec concat_hex36(n :: integer) :: String.t
def concat_hex36(n) do

end
end
```

Erlang Solution:

```
-spec concat_hex36(N :: integer()) -> unicode:unicode_binary().
concat_hex36(N) ->
.
```

Racket Solution:

```
(define/contract (concat-hex36 n)
(-> exact-integer? string?)
)
```