Problem 1271: Hexspeak

Problem Information

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

Problem Description

A decimal number can be converted to its

Hexspeak representation

by first converting it to an uppercase hexadecimal string, then replacing all occurrences of the digit

'0'

with the letter

'O'

, and the digit

'1'

with the letter

'I'

. Such a representation is valid if and only if it consists only of the letters in the set

{'A', 'B', 'C', 'D', 'E', 'F', 'I', 'O'}

.

Given a string

num

representing a decimal integer

n

,

return the

Hexspeak representation

of

n

if it is valid, otherwise return

"ERROR"

.

Example 1:

Input:

num = "257"

Output:

"IOI"

Explanation:

257 is 101 in hexadecimal.

Example 2:

Input:

num = "3"

Output:

"ERROR"

Constraints:

1 <= num.length <= 12

num

does not contain leading zeros.

num represents an integer in the range

[1, 10

12

]

.

Code Snippets

C++:

```
class Solution {
public:
string toHexspeak(string num) {

}
};
```

Java:

```
class Solution {
public String toHexspeak(String num) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function toHexspeak(num: string): string {

};
```

C#:

```
public class Solution {
public string ToHexspeak(string num) {
}
}
```

C:

```
char* toHexspeak(char* num) {

}
```

Go:

```
func toHexspeak(num string) string {

}
```

Kotlin:

```
class Solution {
fun toHexspeak(num: String): String {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
String toHexspeak(String num) {

}
}
```

Scala:

```
object Solution {
def toHexspeak(num: String): String = {

}
}
```

Elixir:

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

end
end
```

Erlang:

```
-spec to_hexspeak(Num :: unicode:unicode_binary()) ->
unicode:unicode_binary().
to_hexspeak(Num) ->
.
```

Racket:

```
(define/contract (to-hexspeak num)
(-> string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Hexspeak
* 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 toHexspeak(string num) {

}
};
```

Java Solution:

```
/**
* Problem: Hexspeak
* 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 toHexspeak(String num) {

}
}
```

Python3 Solution:

```
"""
Problem: Hexspeak
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 toHexspeak(self, num: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Hexspeak
* 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 {string} num
* @return {string}
*/
var toHexspeak = function(num) {

};
```

TypeScript Solution:

```
/**
* Problem: Hexspeak
* 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 toHexspeak(num: string): string {

};
```

C# Solution:

```
/*
* Problem: Hexspeak
* 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 ToHexspeak(string num) {

}
}
```

C Solution:

```
/*
* Problem: Hexspeak
* 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* toHexspeak(char* num) {

}
```

Go Solution:

```
// Problem: Hexspeak
// 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 toHexspeak(num string) string {

}
```

Kotlin Solution:

```
class Solution {
fun toHexspeak(num: String): String {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Hexspeak
// 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 to_hexspeak(num: String) -> String {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {
```

```
/**
* @param String $num
* @return String
*/
function toHexspeak($num) {

}
}
```

Dart Solution:

```
class Solution {
String toHexspeak(String num) {

}
}
```

Scala Solution:

```
object Solution {
def toHexspeak(num: String): String = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

```
-spec to_hexspeak(Num :: unicode:unicode_binary()) ->
unicode:unicode_binary().
to_hexspeak(Num) ->
.
```

Racket Solution:

```
(define/contract (to-hexspeak num)
(-> string? string?)
)
```