Problem 660: Remove 9

Problem Information

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

Problem Description

Start from integer

1

, remove any integer that contains

9

such as

9

,

19

,

29

...

Now, you will have a new integer sequence

[1, 2, 3, 4, 5, 6, 7, 8, 10, 11, ...]

.

Given an integer

n

, return

the

n

th

(

1-indexed

) integer in the new sequence.

Example 1:

Input:

n = 9

Output:

10

Example 2:

Input:

n = 10

Output:

11

Constraints:

1 <= n <= 8 * 10

8

Code Snippets

C++:

```
class Solution {
public:
int newInteger(int n) {

}
};
```

Java:

```
class Solution {
public int newInteger(int n) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function newInteger(n: number): number {

};
```

C#:

```
public class Solution {
public int NewInteger(int n) {

}
}
```

C:

```
int newInteger(int n) {

}
```

Go:

```
func newInteger(n int) int {

}
```

Kotlin:

```
class Solution {
fun newInteger(n: Int): Int {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
int newInteger(int n) {

}
}
```

Scala:

```
object Solution {
def newInteger(n: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec new_integer(n :: integer) :: integer
def new_integer(n) do

end
end
```

Erlang:

```
-spec new_integer(N :: integer()) -> integer().
new_integer(N) ->
.
```

Racket:

```
(define/contract (new-integer n)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Remove 9
* Difficulty: Hard
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/
```

```
class Solution {
public:
int newInteger(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Remove 9
* Difficulty: Hard
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int newInteger(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Remove 9
Difficulty: Hard
Tags: math

Approach: Optimized algorithm based on problem constraints
Time Complexity: O(n) to O(n^2) depending on approach
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def newInteger(self, n: int) -> int:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Remove 9
* Difficulty: Hard
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number} n
* @return {number}
*/
var newInteger = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: Remove 9
* Difficulty: Hard
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
```

```
*/

function newInteger(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Remove 9
* Difficulty: Hard
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int NewInteger(int n) {

}
}
```

C Solution:

```
/*
* Problem: Remove 9
* Difficulty: Hard
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

int newInteger(int n) {

}
```

Go Solution:

```
// Problem: Remove 9
// Difficulty: Hard
// Tags: math
//
// Approach: Optimized algorithm based on problem constraints
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(1) to O(n) depending on approach

func newInteger(n int) int {

}
```

Kotlin Solution:

```
class Solution {
fun newInteger(n: Int): Int {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Remove 9
// Difficulty: Hard
// Tags: math
//
// Approach: Optimized algorithm based on problem constraints
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn new_integer(n: i32) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
int newInteger(int n) {

}
}
```

Scala Solution:

```
object Solution {
def newInteger(n: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec new_integer(n :: integer) :: integer
def new_integer(n) do

end
end
```

Erlang Solution:

```
-spec new_integer(N :: integer()) -> integer().
new_integer(N) ->
.
```

Racket Solution:

```
(define/contract (new-integer n)
(-> exact-integer? exact-integer?)
)
```