Problem 2802: Find The K-th Lucky Number

Problem Information

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

Problem Description

We know that

4

and

7

are

lucky

digits. Also, a number is called

lucky

if it contains

only

lucky digits.

You are given an integer

k

, return

the

k

th

lucky number represented as a

string

.

Example 1:

Input:

k = 4

Output:

"47"

Explanation:

The first lucky number is 4, the second one is 7, the third one is 44 and the fourth one is 47.

Example 2:

Input:

k = 10

Output:

"477"

Explanation:

Here are lucky numbers sorted in increasing order: 4, 7, 44, 47, 74, 77, 444, 447, 474, 477. So the 10

th

lucky number is 477.

Example 3:

Input:

k = 1000

Output:

"777747447"

Explanation:

It can be shown that the 1000

th

lucky number is 777747447.

Constraints:

1 <= k <= 10

9

Code Snippets

C++:

```
class Solution {
public:
string kthLuckyNumber(int k) {
```

```
}
};
```

Java:

```
class Solution {
public String kthLuckyNumber(int k) {

}
}
```

Python3:

```
class Solution:
def kthLuckyNumber(self, k: int) -> str:
```

Python:

```
class Solution(object):
def kthLuckyNumber(self, k):
"""
:type k: int
:rtype: str
"""
```

JavaScript:

```
/**
* @param {number} k
* @return {string}
*/
var kthLuckyNumber = function(k) {

};
```

TypeScript:

```
function kthLuckyNumber(k: number): string {

};
```

C#:

```
public class Solution {
public string KthLuckyNumber(int k) {

}
}
```

C:

```
char* kthLuckyNumber(int k) {

}
```

Go:

```
func kthLuckyNumber(k int) string {

}
```

Kotlin:

```
class Solution {
fun kthLuckyNumber(k: Int): String {

}
}
```

Swift:

```
class Solution {
func kthLuckyNumber(_ k: Int) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn kth_lucky_number(k: i32) -> String {

}
}
```

Ruby:

```
# @param {Integer} k
# @return {String}
def kth_lucky_number(k)

end
```

PHP:

```
class Solution {

/**
* @param Integer $k
* @return String
*/
function kthLuckyNumber($k) {

}
}
```

Dart:

```
class Solution {
String kthLuckyNumber(int k) {

}
}
```

Scala:

```
object Solution {
def kthLuckyNumber(k: Int): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec kth_lucky_number(k :: integer) :: String.t
def kth_lucky_number(k) do
```

```
end
end
```

Erlang:

```
-spec kth_lucky_number(K :: integer()) -> unicode:unicode_binary().
kth_lucky_number(K) ->
.
```

Racket:

```
(define/contract (kth-lucky-number k)
(-> exact-integer? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find The K-th Lucky Number
* Difficulty: Medium
* Tags: string, math, sort
*
* 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 kthLuckyNumber(int k) {

}
};
```

Java Solution:

```
/**
* Problem: Find The K-th Lucky Number
```

```
* Difficulty: Medium
* Tags: string, math, sort
*
* 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 kthLuckyNumber(int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Find The K-th Lucky Number
Difficulty: Medium
Tags: string, math, sort

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

Python Solution:

```
class Solution(object):
def kthLuckyNumber(self, k):
"""
:type k: int
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Find The K-th Lucky Number
* Difficulty: Medium
* Tags: string, math, sort
*
* 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} k
* @return {string}
*/
var kthLuckyNumber = function(k) {

};
```

TypeScript Solution:

```
/**
* Problem: Find The K-th Lucky Number
* Difficulty: Medium
* Tags: string, math, sort
*
* 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 kthLuckyNumber(k: number): string {

};
```

C# Solution:

```
/*
* Problem: Find The K-th Lucky Number
* Difficulty: Medium
* Tags: string, math, sort
*
* 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 KthLuckyNumber(int k) {

}
}
```

C Solution:

```
/*
* Problem: Find The K-th Lucky Number
* Difficulty: Medium
* Tags: string, math, sort
*
* 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* kthLuckyNumber(int k) {

}
```

Go Solution:

```
// Problem: Find The K-th Lucky Number
// Difficulty: Medium
// Tags: string, math, sort
//
// 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 kthLuckyNumber(k int) string {

}
```

Kotlin Solution:

```
class Solution {
fun kthLuckyNumber(k: Int): String {

}
}
```

Swift Solution:

```
class Solution {
func kthLuckyNumber(_ k: Int) -> String {

}
}
```

Rust Solution:

```
// Problem: Find The K-th Lucky Number
// Difficulty: Medium
// Tags: string, math, sort
//
// 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 kth_lucky_number(k: i32) -> String {

}
}
```

Ruby Solution:

```
# @param {Integer} k
# @return {String}
def kth_lucky_number(k)

end
```

PHP Solution:

```
class Solution {
```

```
/**
* @param Integer $k
* @return String
*/
function kthLuckyNumber($k) {

}
}
```

Dart Solution:

```
class Solution {
String kthLuckyNumber(int k) {

}
}
```

Scala Solution:

```
object Solution {
def kthLuckyNumber(k: Int): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec kth_lucky_number(k :: integer) :: String.t
def kth_lucky_number(k) do

end
end
```

Erlang Solution:

```
-spec kth_lucky_number(K :: integer()) -> unicode:unicode_binary().
kth_lucky_number(K) ->
.
```

Racket Solution:

```
(define/contract (kth-lucky-number k)
(-> exact-integer? string?)
)
```