Problem 1015: Smallest Integer Divisible by K

Problem Information

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

Problem Description

Given a positive integer

k

, you need to find the

length

of the

smallest

positive integer

n

such that

n

is divisible by

k

, and

n

only contains the digit

1

.

Return

the

length

of

n

. If there is no such

n

, return -1.

Note:

n

may not fit in a 64-bit signed integer.

Example 1:

Input:

k = 1

Output:

1

Explanation:

The smallest answer is n = 1, which has length 1.

Example 2:

Input:

k = 2

Output:

-1

Explanation:

There is no such positive integer n divisible by 2.

Example 3:

Input:

k = 3

Output:

3

Explanation:

The smallest answer is n = 111, which has length 3.

Constraints:

1 <= k <= 10

5

Code Snippets

C++:

```
class Solution {
public:
int smallestRepunitDivByK(int k) {

}
};
```

Java:

```
class Solution {
public int smallestRepunitDivByK(int k) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function smallestRepunitDivByK(k: number): number {

};
```

C#:

```
public class Solution {
public int SmallestRepunitDivByK(int k) {

}
}
```

C:

```
int smallestRepunitDivByK(int k) {

}
```

Go:

```
func smallestRepunitDivByK(k int) int {

}
```

Kotlin:

```
class Solution {
fun smallestRepunitDivByK(k: Int): Int {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
int smallestRepunitDivByK(int k) {

}
}
```

Scala:

```
object Solution {
def smallestRepunitDivByK(k: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec smallest_repunit_div_by_k(k :: integer) :: integer
def smallest_repunit_div_by_k(k) do

end
end
```

Erlang:

```
-spec smallest_repunit_div_by_k(K :: integer()) -> integer().
smallest_repunit_div_by_k(K) ->
.
```

Racket:

```
(define/contract (smallest-repunit-div-by-k k)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Smallest Integer Divisible by K
* Difficulty: Medium
* Tags: math, hash
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
int smallestRepunitDivByK(int k) {
}
};
```

Java Solution:

```
/**
* Problem: Smallest Integer Divisible by K
* Difficulty: Medium
* Tags: math, hash
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

class Solution {
public int smallestRepunitDivByK(int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Smallest Integer Divisible by K
Difficulty: Medium
Tags: math, hash

Approach: Use hash map for O(1) lookups
Time Complexity: O(n) to O(n^2) depending on approach
Space Complexity: O(n) for hash map
"""

class Solution:
def smallestRepunitDivByK(self, k: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Smallest Integer Divisible by K
* Difficulty: Medium
* Tags: math, hash
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

/**
* @param {number} k
* @return {number}
*/
var smallestRepunitDivByK = function(k) {

};
```

TypeScript Solution:

```
/**
* Problem: Smallest Integer Divisible by K
* Difficulty: Medium
* Tags: math, hash
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

function smallestRepunitDivByK(k: number): number {

};
```

C# Solution:

```
/*
* Problem: Smallest Integer Divisible by K
* Difficulty: Medium
* Tags: math, hash
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int SmallestRepunitDivByK(int k) {

}
}
```

C Solution:

```
/*
* Problem: Smallest Integer Divisible by K
* Difficulty: Medium
* Tags: math, hash
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

int smallestRepunitDivByK(int k) {

}
```

Go Solution:

```
// Problem: Smallest Integer Divisible by K
// Difficulty: Medium
// Tags: math, hash
//
// Approach: Use hash map for O(1) lookups
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(n) for hash map

func smallestRepunitDivByK(k int) int {

}
```

Kotlin Solution:

```
class Solution {
fun smallestRepunitDivByK(k: Int): Int {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Smallest Integer Divisible by K
// Difficulty: Medium
// Tags: math, hash
//
// Approach: Use hash map for O(1) lookups
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(n) for hash map

impl Solution {
pub fn smallest_repunit_div_by_k(k: i32) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
int smallestRepunitDivByK(int k) {

}
}
```

Scala Solution:

```
object Solution {
def smallestRepunitDivByK(k: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec smallest_repunit_div_by_k(k :: integer) :: integer
def smallest_repunit_div_by_k(k) do

end
```

```

end
```

Erlang Solution:

```
-spec smallest_repunit_div_by_k(K :: integer()) -> integer().
smallest_repunit_div_by_k(K) ->
.
```

Racket Solution:

```
(define/contract (smallest-repunit-div-by-k k)
(-> exact-integer? exact-integer?)
)
```