Problem 989: Add to Array-Form of Integer

Problem Information

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

Problem Description

The

array-form

of an integer

num

is an array representing its digits in left to right order.

For example, for

num = 1321

, the array form is

[1,3,2,1]

.

Given

num

, the

array-form

of an integer, and an integer

k

, return

the

array-form

of the integer

num + k

.

Example 1:

Input:

num = [1,2,0,0], k = 34

Output:

[1,2,3,4]

Explanation:

1200 + 34 = 1234

Example 2:

Input:

num = [2,7,4], k = 181

Output:

[4,5,5]

Explanation:

274 + 181 = 455

Example 3:

Input:

num = [2,1,5], k = 806

Output:

[1,0,2,1]

Explanation:

215 + 806 = 1021

Constraints:

1 <= num.length <= 10

4

0 <= num[i] <= 9

num

does not contain any leading zeros except for the zero itself.

1 <= k <= 10

4

Code Snippets

C++:

```
class Solution {
public:
vector<int> addToArrayForm(vector<int>& num, int k) {

}
};
```

Java:

```
class Solution {
public List<Integer> addToArrayForm(int[] num, int k) {

}
}
```

Python3:

```
class Solution:
def addToArrayForm(self, num: List[int], k: int) -> List[int]:
```

Python:

```
class Solution(object):
def addToArrayForm(self, num, k):
"""
:type num: List[int]
:type k: int
:rtype: List[int]
"""
```

JavaScript:

```
/**
* @param {number[]} num
* @param {number} k
* @return {number[]}
*/
var addToArrayForm = function(num, k) {

};
```

TypeScript:

```
function addToArrayForm(num: number[], k: number): number[] {

};
```

C#:

```
public class Solution {
public IList<int> AddToArrayForm(int[] num, int k) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* addToArrayForm(int* num, int numSize, int k, int* returnSize) {

}
```

Go:

```
func addToArrayForm(num []int, k int) []int {

}
```

Kotlin:

```
class Solution {
fun addToArrayForm(num: IntArray, k: Int): List<Int> {

}
}
```

Swift:

```
class Solution {
func addToArrayForm(_ num: [Int], _ k: Int) -> [Int] {

}
```

```
}
```

Rust:

```
impl Solution {
pub fn add_to_array_form(num: Vec<i32>, k: i32) -> Vec<i32> {

}
}
```

Ruby:

```
# @param {Integer[]} num
# @param {Integer} k
# @return {Integer[]}
def add_to_array_form(num, k)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $num
* @param Integer $k
* @return Integer[]
*/
function addToArrayForm($num, $k) {

}
}
```

Dart:

```
class Solution {
List<int> addToArrayForm(List<int> num, int k) {

}
}
```

Scala:

```
object Solution {
def addToArrayForm(num: Array[Int], k: Int): List[Int] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec add_to_array_form(num :: [integer], k :: integer) :: [integer]
def add_to_array_form(num, k) do

end
end
```

Erlang:

```
-spec add_to_array_form(Num :: [integer()], K :: integer()) -> [integer()].
add_to_array_form(Num, K) ->
.
```

Racket:

```
(define/contract (add-to-array-form num k)
(-> (listof exact-integer?) exact-integer? (listof exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Add to Array-Form of Integer
* Difficulty: Easy
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/
```

```
class Solution {
public:
vector<int> addToArrayForm(vector<int>& num, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Add to Array-Form of Integer
* Difficulty: Easy
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public List<Integer> addToArrayForm(int[] num, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Add to Array-Form of Integer
Difficulty: Easy
Tags: array, math

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def addToArrayForm(self, num: List[int], k: int) -> List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def addToArrayForm(self, num, k):
"""
:type num: List[int]
:type k: int
:rtype: List[int]
"""
```

JavaScript Solution:

```
/**
* Problem: Add to Array-Form of Integer
* Difficulty: Easy
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number[]} num
* @param {number} k
* @return {number[]}
*/
var addToArrayForm = function(num, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Add to Array-Form of Integer
* Difficulty: Easy
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/
```

```
function addToArrayForm(num: number[], k: number): number[] {

};
```

C# Solution:

```
/*
* Problem: Add to Array-Form of Integer
* Difficulty: Easy
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public IList<int> AddToArrayForm(int[] num, int k) {

}
}
```

C Solution:

```
/*
* Problem: Add to Array-Form of Integer
* Difficulty: Easy
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* addToArrayForm(int* num, int numSize, int k, int* returnSize) {

}
```

Go Solution:

```
// Problem: Add to Array-Form of Integer
// Difficulty: Easy
// Tags: array, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func addToArrayForm(num []int, k int) []int {

}
```

Kotlin Solution:

```
class Solution {
fun addToArrayForm(num: IntArray, k: Int): List<Int> {

}
}
```

Swift Solution:

```
class Solution {
func addToArrayForm(_ num: [Int], _ k: Int) -> [Int] {

}
}
```

Rust Solution:

```
// Problem: Add to Array-Form of Integer
// Difficulty: Easy
// Tags: array, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn add_to_array_form(num: Vec<i32>, k: i32) -> Vec<i32> {
```

```
}
}
```

Ruby Solution:

```
# @param {Integer[]} num
# @param {Integer} k
# @return {Integer[]}
def add_to_array_form(num, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $num
* @param Integer $k
* @return Integer[]
*/
function addToArrayForm($num, $k) {

}
}
```

Dart Solution:

```
class Solution {
List<int> addToArrayForm(List<int> num, int k) {

}
}
```

Scala Solution:

```
object Solution {
def addToArrayForm(num: Array[Int], k: Int): List[Int] = {

}
```

```
}
```

Elixir Solution:

```
defmodule Solution do
@spec add_to_array_form(num :: [integer], k :: integer) :: [integer]
def add_to_array_form(num, k) do

end
end
```

Erlang Solution:

```
-spec add_to_array_form(Num :: [integer()], K :: integer()) -> [integer()].
add_to_array_form(Num, K) ->
.
```

Racket Solution:

```
(define/contract (add-to-array-form num k)
(-> (listof exact-integer?) exact-integer? (listof exact-integer?))
)
```