Problem 1881: Maximum Value after Insertion

Problem Information

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

Problem Description

You are given a very large integer

n

, represented as a string,■■■■■■ and an integer digit

x

. The digits in

n

and the digit

x

are in the

inclusive

range

[1, 9]

, and

n

may represent a

negative

number.

You want to

maximize

n

's numerical value

by inserting

x

anywhere in the decimal representation of

n

. You

cannot

insert

x

to the left of the negative sign.

For example, if

n = 73

and

x = 6

, it would be best to insert it between

7

and

3

, making

n = 763

.

If

n = -55

and

x = 2

, it would be best to insert it before the first

5

, making

n = -255

.

Return

a string representing the

maximum

value of

n

after the insertion

.

Example 1:

Input:

n = "99", x = 9

Output:

"999"

Explanation:

The result is the same regardless of where you insert 9.

Example 2:

Input:

n = "-13", x = 2

Output:

"-123"

Explanation:

You can make n one of {-213, -123, -132}, and the largest of those three is -123.

Constraints:

1 <= n.length <= 10

5

1 <= x <= 9

The digits in

n

are in the range

[1, 9]

.

n

is a valid representation of an integer.

In the case of a negative

n

,■■■■■■ it will begin with

'-'

.

Code Snippets

C++:

```
class Solution {
public:
string maxValue(string n, int x) {
}
};
```

Java:

```
class Solution {
public String maxValue(String n, int x) {

}
}
```

Python3:

```
class Solution:
def maxValue(self, n: str, x: int) -> str:
```

Python:

```
class Solution(object):
def maxValue(self, n, x):
"""
:type n: str
:type x: int
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} n
* @param {number} x
* @return {string}
*/
var maxValue = function(n, x) {

};
```

TypeScript:

```
function maxValue(n: string, x: number): string {

};
```

C#:

```
public class Solution {
public string MaxValue(string n, int x) {

}
}
```

C:

```
char* maxValue(char* n, int x) {

}
```

Go:

```
func maxValue(n string, x int) string {

}
```

Kotlin:

```
class Solution {
fun maxValue(n: String, x: Int): String {

}
}
```

Swift:

```
class Solution {
func maxValue(_ n: String, _ x: Int) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn max_value(n: String, x: i32) -> String {

}
}
```

Ruby:

```
# @param {String} n
# @param {Integer} x
# @return {String}
def max_value(n, x)

end
```

PHP:

```
class Solution {

/**
* @param String $n
* @param Integer $x
* @return String
*/
function maxValue($n, $x) {

}
}
```

Dart:

```
class Solution {
String maxValue(String n, int x) {

}
}
```

Scala:

```
object Solution {
def maxValue(n: String, x: Int): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec max_value(n :: String.t, x :: integer) :: String.t
```

```
def max_value(n, x) do

end
end
```

Erlang:

```
-spec max_value(N :: unicode:unicode_binary(), X :: integer()) ->
unicode:unicode_binary().
max_value(N, X) ->
.
```

Racket:

```
(define/contract (max-value n x)
(-> string? exact-integer? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Value after Insertion
* Difficulty: Medium
* Tags: string, greedy
*
* 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 maxValue(string n, int x) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Value after Insertion
* Difficulty: Medium
* Tags: string, greedy
*
* 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 maxValue(String n, int x) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Value after Insertion
Difficulty: Medium
Tags: string, greedy

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 maxValue(self, n: str, x: int) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def maxValue(self, n, x):
"""
:type n: str
:type x: int
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Maximum Value after Insertion
* Difficulty: Medium
* Tags: string, greedy
*
* 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} n
* @param {number} x
* @return {string}
*/
var maxValue = function(n, x) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Value after Insertion
* Difficulty: Medium
* Tags: string, greedy
*
* 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 maxValue(n: string, x: number): string {

};
```

C# Solution:

```
/*
* Problem: Maximum Value after Insertion
* Difficulty: Medium
* Tags: string, greedy
*
* 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 MaxValue(string n, int x) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Value after Insertion
* Difficulty: Medium
* Tags: string, greedy
*
* 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* maxValue(char* n, int x) {

}
```

Go Solution:

```
// Problem: Maximum Value after Insertion
// Difficulty: Medium
// Tags: string, greedy
//
// 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 maxValue(n string, x int) string {
}
```

Kotlin Solution:

```
class Solution {
fun maxValue(n: String, x: Int): String {

}
}
```

Swift Solution:

```
class Solution {
func maxValue(_ n: String, _ x: Int) -> String {

}
}
```

Rust Solution:

```
// Problem: Maximum Value after Insertion
// Difficulty: Medium
// Tags: string, greedy
//
// 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 max_value(n: String, x: i32) -> String {

}
}
```

Ruby Solution:

```
# @param {String} n
# @param {Integer} x
# @return {String}
def max_value(n, x)
```

```
end
```

PHP Solution:

```
class Solution {

/**
* @param String $n
* @param Integer $x
* @return String
*/
function maxValue($n, $x) {

}
}
```

Dart Solution:

```
class Solution {
String maxValue(String n, int x) {

}
}
```

Scala Solution:

```
object Solution {
def maxValue(n: String, x: Int): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec max_value(n :: String.t, x :: integer) :: String.t
def max_value(n, x) do

end
end
```

Erlang Solution:

```
-spec max_value(N :: unicode:unicode_binary(), X :: integer()) ->
unicode:unicode_binary().
max_value(N, X) ->
.
```

Racket Solution:

```
(define/contract (max-value n x)
(-> string? exact-integer? string?)
)
```