Problem 1323: Maximum 69 Number

Problem Information

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

Problem Description

You are given a positive integer

num

consisting only of digits

6

and

9

.

Return

the maximum number you can get by changing

at most

one digit (

6

becomes

9

, and

9

becomes

6

)

.

Example 1:

Input:

num = 9669

Output:

9969

Explanation:

Changing the first digit results in 6669. Changing the second digit results in 9969. Changing the third digit results in 9699. Changing the fourth digit results in 9666. The maximum number is 9969.

Example 2:

Input:

num = 9996

Output:

9999

Explanation:

Changing the last digit 6 to 9 results in the maximum number.

Example 3:

Input:

num = 9999

Output:

9999

Explanation:

It is better not to apply any change.

Constraints:

1 <= num <= 10

4

num

consists of only

6

and

9

digits.

Code Snippets

C++:

```
class Solution {
public:
int maximum69Number (int num) {

}
};
```

Java:

```
class Solution {
public int maximum69Number (int num) {

}
}
```

Python3:

```
class Solution:
def maximum69Number (self, num: int) -> int:
```

Python:

```
class Solution(object):
def maximum69Number (self, num):
"""
:type num: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} num
* @return {number}
*/
var maximum69Number = function(num) {

};
```

TypeScript:

```
function maximum69Number (num: number): number {

};
```

C#:

```
public class Solution {
public int Maximum69Number (int num) {

}
}
```

C:

```
int maximum69Number (int num) {

}
```

Go:

```
func maximum69Number (num int) int {

}
```

Kotlin:

```
class Solution {
fun maximum69Number (num: Int): Int {

}
}
```

Swift:

```
class Solution {
func maximum69Number (_ num: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn maximum69_number (num: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} num
# @return {Integer}
def maximum69_number (num)

end
```

PHP:

```
class Solution {

/**
* @param Integer $num
* @return Integer
*/
function maximum69Number ($num) {

}
}
```

Dart:

```
class Solution {
int maximum69Number (int num) {

}
}
```

Scala:

```
object Solution {
def maximum69Number (num: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec maximum69_number (num :: integer) :: integer
def maximum69_number (num) do

end
end
```

Erlang:

```
-spec maximum69_number (Num :: integer()) -> integer().
maximum69_number (Num) ->
.
```

Racket:

```
(define/contract (maximum69-number num)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum 69 Number
* Difficulty: Easy
* Tags: greedy, math
*
* Approach: Greedy algorithm with local optimal choices
* 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 maximum69Number (int num) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum 69 Number
* Difficulty: Easy
* Tags: greedy, math
*
* Approach: Greedy algorithm with local optimal choices
* 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 maximum69Number (int num) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum 69 Number
Difficulty: Easy
Tags: greedy, math

Approach: Greedy algorithm with local optimal choices
Time Complexity: O(n) to O(n^2) depending on approach
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def maximum69Number (self, num: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def maximum69Number (self, num):
"""
:type num: int
:rtype: int
```

```
"""
```

JavaScript Solution:

```
/**
* Problem: Maximum 69 Number
* Difficulty: Easy
* Tags: greedy, math
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number} num
* @return {number}
*/
var maximum69Number = function(num) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum 69 Number
* Difficulty: Easy
* Tags: greedy, math
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

function maximum69Number (num: number): number {

};
```

C# Solution:

```
/*
* Problem: Maximum 69 Number
* Difficulty: Easy
* Tags: greedy, math
*
* Approach: Greedy algorithm with local optimal choices
* 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 Maximum69Number (int num) {

}
}
```

C Solution:

```
/*
* Problem: Maximum 69 Number
* Difficulty: Easy
* Tags: greedy, math
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

int maximum69Number (int num) {

}
```

Go Solution:

```
// Problem: Maximum 69 Number
// Difficulty: Easy
// Tags: greedy, math
//
// Approach: Greedy algorithm with local optimal choices
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(1) to O(n) depending on approach
```

```
func maximum69Number (num int) int {

}
```

Kotlin Solution:

```
class Solution {
fun maximum69Number (num: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func maximum69Number (_ num: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Maximum 69 Number
// Difficulty: Easy
// Tags: greedy, math
//
// Approach: Greedy algorithm with local optimal choices
// 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 maximum69_number (num: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} num
# @return {Integer}
def maximum69_number (num)
```

```
end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $num
* @return Integer
*/
function maximum69Number ($num) {

}
}
```

Dart Solution:

```
class Solution {
int maximum69Number (int num) {

}
}
```

Scala Solution:

```
object Solution {
def maximum69Number (num: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec maximum69_number (num :: integer) :: integer
def maximum69_number (num) do

end
end
```

Erlang Solution:

```
-spec maximum69_number (Num :: integer()) -> integer().
maximum69_number (Num) ->
.
```

Racket Solution:

```
(define/contract (maximum69-number num)
(-> exact-integer? exact-integer?)
)
```