Problem 2384: Largest Palindromic Number

Problem Information

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

Problem Description

You are given a string

num

consisting of digits only.

Return

the

largest palindromic

integer (in the form of a string) that can be formed using digits taken from

num

. It should not contain

leading zeroes

.

Notes:

You do

not

need to use all the digits of

num

, but you must use

at least

one digit.

The digits can be reordered.

Example 1:

Input:

num = "444947137"

Output:

"7449447"

Explanation:

Use the digits "4449477" from "

44494

7

13

7

" to form the palindromic integer "7449447". It can be shown that "7449447" is the largest palindromic integer that can be formed.

Example 2:

Input:

num = "00009"

Output:

"9"

Explanation:

It can be shown that "9" is the largest palindromic integer that can be formed. Note that the integer returned should not contain leading zeroes.

Constraints:

1 <= num.length <= 10

5

num

consists of digits.

Code Snippets

C++:

```
class Solution {
public:
string largestPalindromic(string num) {

}
};
```

Java:

```
class Solution {
public String largestPalindromic(String num) {

}
}
```

Python3:

```
class Solution:
def largestPalindromic(self, num: str) -> str:
```

Python:

```
class Solution(object):
def largestPalindromic(self, num):
"""
:type num: str
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} num
* @return {string}
*/
var largestPalindromic = function(num) {

};
```

TypeScript:

```
function largestPalindromic(num: string): string {

};
```

C#:

```
public class Solution {
public string LargestPalindromic(string num) {

}
}
```

C:

```
char* largestPalindromic(char* num) {

}
```

Go:

```
func largestPalindromic(num string) string {

}
```

Kotlin:

```
class Solution {
fun largestPalindromic(num: String): String {

}
}
```

Swift:

```
class Solution {
func largestPalindromic(_ num: String) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn largest_palindromic(num: String) -> String {

}
}
```

Ruby:

```
# @param {String} num
# @return {String}
def largest_palindromic(num)

end
```

PHP:

```
class Solution {

/**
* @param String $num
* @return String
*/
function largestPalindromic($num) {

}
}
```

Dart:

```
class Solution {
String largestPalindromic(String num) {

}
}
```

Scala:

```
object Solution {
def largestPalindromic(num: String): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec largest_palindromic(num :: String.t) :: String.t
def largest_palindromic(num) do

end
end
```

Erlang:

```
-spec largest_palindromic(Num :: unicode:unicode_binary()) ->
unicode:unicode_binary().
largest_palindromic(Num) ->
```

```
.
```

Racket:

```
(define/contract (largest-palindromic num)
(-> string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Largest Palindromic Number
* Difficulty: Medium
* Tags: string, greedy, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
string largestPalindromic(string num) {

}
};
```

Java Solution:

```
/**
* Problem: Largest Palindromic Number
* Difficulty: Medium
* Tags: string, greedy, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/
```

```
class Solution {
public String largestPalindromic(String num) {

}
}
```

Python3 Solution:

```
"""
Problem: Largest Palindromic Number
Difficulty: Medium
Tags: string, greedy, hash

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

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

Python Solution:

```
class Solution(object):
def largestPalindromic(self, num):
"""
:type num: str
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Largest Palindromic Number
* Difficulty: Medium
* Tags: string, greedy, hash
*
* Approach: String manipulation with hash map or two pointers
```

```
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {string} num
* @return {string}
*/
var largestPalindromic = function(num) {

};
```

TypeScript Solution:

```
/**
* Problem: Largest Palindromic Number
* Difficulty: Medium
* Tags: string, greedy, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function largestPalindromic(num: string): string {

};
```

C# Solution:

```
/*
* Problem: Largest Palindromic Number
* Difficulty: Medium
* Tags: string, greedy, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
```

```
public string LargestPalindromic(string num) {

}
}
```

C Solution:

```
/*
* Problem: Largest Palindromic Number
* Difficulty: Medium
* Tags: string, greedy, hash
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

char* largestPalindromic(char* num) {

}
```

Go Solution:

```
// Problem: Largest Palindromic Number
// Difficulty: Medium
// Tags: string, greedy, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func largestPalindromic(num string) string {

}
```

Kotlin Solution:

```
class Solution {
fun largestPalindromic(num: String): String {

}
```

```
}
```

Swift Solution:

```
class Solution {
func largestPalindromic(_ num: String) -> String {

}
}
```

Rust Solution:

```
// Problem: Largest Palindromic Number
// Difficulty: Medium
// Tags: string, greedy, hash
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn largest_palindromic(num: String) -> String {

}
}
```

Ruby Solution:

```
# @param {String} num
# @return {String}
def largest_palindromic(num)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $num
* @return String
```

```
*/
function largestPalindromic($num) {

}
}
```

Dart Solution:

```
class Solution {
String largestPalindromic(String num) {

}
}
```

Scala Solution:

```
object Solution {
def largestPalindromic(num: String): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec largest_palindromic(num :: String.t) :: String.t
def largest_palindromic(num) do

end
end
```

Erlang Solution:

```
-spec largest_palindromic(Num :: unicode:unicode_binary()) ->
unicode:unicode_binary().
largest_palindromic(Num) ->
.
```

Racket Solution:

```
(define/contract (largest-palindromic num)
(-> string? string?)
)
```