Problem 1405: Longest Happy String

Problem Information

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

Problem Description

A string

s

is called

happy

if it satisfies the following conditions:

s

only contains the letters

'a'

,

'b'

, and

'c'

.

s

does not contain any of

"aaa"

,

"bbb"

, or

"ccc"

as a substring.

s

contains

at most

a

occurrences of the letter

'a'

.

s

contains

at most

b

occurrences of the letter

'b'

.

s

contains

at most

c

occurrences of the letter

'c'

.

Given three integers

a

,

b

, and

c

, return

the

longest possible happy

string

. If there are multiple longest happy strings, return

any of them

. If there is no such string, return

the empty string

""

.

A

substring

is a contiguous sequence of characters within a string.

Example 1:

Input:

a = 1, b = 1, c = 7

Output:

"ccaccbcc"

Explanation:

"ccbccacc" would also be a correct answer.

Example 2:

Input:

a = 7, b = 1, c = 0

Output:

"aabaa"

Explanation:

It is the only correct answer in this case.

Constraints:

0 <= a, b, c <= 100

a + b + c > 0

Code Snippets

C++:

```
class Solution {
public:
string longestDiverseString(int a, int b, int c) {

}
};
```

Java:

```
class Solution {
public String longestDiverseString(int a, int b, int c) {

}
}
```

Python3:

```
class Solution:
def longestDiverseString(self, a: int, b: int, c: int) -> str:
```

Python:

```
class Solution(object):
def longestDiverseString(self, a, b, c):
```

```
"""
:type a: int
:type b: int
:type c: int
:rtype: str
"""
```

JavaScript:

```
/**
* @param {number} a
* @param {number} b
* @param {number} c
* @return {string}
*/
var longestDiverseString = function(a, b, c) {

};
```

TypeScript:

```
function longestDiverseString(a: number, b: number, c: number): string {

};
```

C#:

```
public class Solution {
public string LongestDiverseString(int a, int b, int c) {

}
}
```

C:

```
char* longestDiverseString(int a, int b, int c) {

}
```

Go:

```
func longestDiverseString(a int, b int, c int) string {

}
```

Kotlin:

```
class Solution {
fun longestDiverseString(a: Int, b: Int, c: Int): String {

}
}
```

Swift:

```
class Solution {
func longestDiverseString(_ a: Int, _ b: Int, _ c: Int) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn longest_diverse_string(a: i32, b: i32, c: i32) -> String {

}
}
```

Ruby:

```
# @param {Integer} a
# @param {Integer} b
# @param {Integer} c
# @return {String}
def longest_diverse_string(a, b, c)

end
```

PHP:

```
class Solution {

/**
```

```
* @param Integer $a
* @param Integer $b
* @param Integer $c
* @return String
*/
function longestDiverseString($a, $b, $c) {

}
}
```

Dart:

```
class Solution {
String longestDiverseString(int a, int b, int c) {

}
}
```

Scala:

```
object Solution {
def longestDiverseString(a: Int, b: Int, c: Int): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec longest_diverse_string(a :: integer, b :: integer, c :: integer) ::
String.t
def longest_diverse_string(a, b, c) do

end
end
```

Erlang:

```
-spec longest_diverse_string(A :: integer(), B :: integer(), C :: integer())
-> unicode:unicode_binary().
longest_diverse_string(A, B, C) ->
.
```

Racket:

```
(define/contract (longest-diverse-string a b c)
(-> exact-integer? exact-integer? exact-integer? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Longest Happy String
* Difficulty: Medium
* Tags: string, tree, greedy, queue, heap
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public:
string longestDiverseString(int a, int b, int c) {

}
};
```

Java Solution:

```
/**
* Problem: Longest Happy String
* Difficulty: Medium
* Tags: string, tree, greedy, queue, heap
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public String longestDiverseString(int a, int b, int c) {
```

```
}
}
```

Python3 Solution:

```
"""
Problem: Longest Happy String
Difficulty: Medium
Tags: string, tree, greedy, queue, heap

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

class Solution:
def longestDiverseString(self, a: int, b: int, c: int) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def longestDiverseString(self, a, b, c):
"""
:type a: int
:type b: int
:type c: int
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Longest Happy String
* Difficulty: Medium
* Tags: string, tree, greedy, queue, heap
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
```

```
* Space Complexity: O(h) for recursion stack where h is height
*/

/**
* @param {number} a
* @param {number} b
* @param {number} c
* @return {string}
*/
var longestDiverseString = function(a, b, c) {

};
```

TypeScript Solution:

```
/**
* Problem: Longest Happy String
* Difficulty: Medium
* Tags: string, tree, greedy, queue, heap
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

function longestDiverseString(a: number, b: number, c: number): string {

};
```

C# Solution:

```
/*
* Problem: Longest Happy String
* Difficulty: Medium
* Tags: string, tree, greedy, queue, heap
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/
```

```
public class Solution {
public string LongestDiverseString(int a, int b, int c) {

}
}
```

C Solution:

```
/*
* Problem: Longest Happy String
* Difficulty: Medium
* Tags: string, tree, greedy, queue, heap
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

char* longestDiverseString(int a, int b, int c) {

}
```

Go Solution:

```
// Problem: Longest Happy String
// Difficulty: Medium
// Tags: string, tree, greedy, queue, heap
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func longestDiverseString(a int, b int, c int) string {

}
```

Kotlin Solution:

```
class Solution {
fun longestDiverseString(a: Int, b: Int, c: Int): String {
```

```
}
}
```

Swift Solution:

```
class Solution {
func longestDiverseString(_ a: Int, _ b: Int, _ c: Int) -> String {

}
}
```

Rust Solution:

```
// Problem: Longest Happy String
// Difficulty: Medium
// Tags: string, tree, greedy, queue, heap
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

impl Solution {
pub fn longest_diverse_string(a: i32, b: i32, c: i32) -> String {

}
}
```

Ruby Solution:

```
# @param {Integer} a
# @param {Integer} b
# @param {Integer} c
# @return {String}
def longest_diverse_string(a, b, c)

end
```

PHP Solution:

```
class Solution {
```

```
/**
* @param Integer $a
* @param Integer $b
* @param Integer $c
* @return String
*/
function longestDiverseString($a, $b, $c) {

}
}
```

Dart Solution:

```
class Solution {
String longestDiverseString(int a, int b, int c) {

}
}
```

Scala Solution:

```
object Solution {
def longestDiverseString(a: Int, b: Int, c: Int): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec longest_diverse_string(a :: integer, b :: integer, c :: integer) ::
String.t
def longest_diverse_string(a, b, c) do

end
end
```

Erlang Solution:

```
-spec longest_diverse_string(A :: integer(), B :: integer(), C :: integer())
-> unicode:unicode_binary().
```

```
longest_diverse_string(A, B, C) ->
.
```

Racket Solution:

```
(define/contract (longest-diverse-string a b c)
(-> exact-integer? exact-integer? exact-integer? string?)
)
```