Problem 1507: Reformat Date

Problem Information

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

Problem Description

Given a

date

string in the form

Day Month Year

, where:

Day

is in the set

{"1st", "2nd", "3rd", "4th", ..., "30th", "31st"}

.

Month

is in the set

{"Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"}

.

Year

is in the range

[1900, 2100]

.

Convert the date string to the format

YYYY-MM-DD

, where:

YYYY

denotes the 4 digit year.

MM

denotes the 2 digit month.

DD

denotes the 2 digit day.

Example 1:

Input:

date = "20th Oct 2052"

Output:

"2052-10-20"

Example 2:

Input:

date = "6th Jun 1933"

Output:

"1933-06-06"

Example 3:

Input:

date = "26th May 1960"

Output:

"1960-05-26"

Constraints:

The given dates are guaranteed to be valid, so no error handling is necessary.

Code Snippets

C++:

```
class Solution {
public:
string reformatDate(string date) {

}
};
```

Java:

```
class Solution {
public String reformatDate(String date) {

}
}
```

Python3:

```
class Solution:
def reformatDate(self, date: str) -> str:
```

Python:

```
class Solution(object):
def reformatDate(self, date):
"""
:type date: str
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} date
* @return {string}
*/
var reformatDate = function(date) {

};
```

TypeScript:

```
function reformatDate(date: string): string {

};
```

C#:

```
public class Solution {
public string ReformatDate(string date) {

}
}
```

C:

```
char* reformatDate(char* date) {

}
```

Go:

```
func reformatDate(date string) string {

}
```

Kotlin:

```
class Solution {
fun reformatDate(date: String): String {

}
}
```

Swift:

```
class Solution {
func reformatDate(_ date: String) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn reformat_date(date: String) -> String {

}
}
```

Ruby:

```
# @param {String} date
# @return {String}
def reformat_date(date)

end
```

PHP:

```
class Solution {

/**
* @param String $date
* @return String
*/
function reformatDate($date) {

}
}
```

Dart:

```
class Solution {
String reformatDate(String date) {

}
}
```

Scala:

```
object Solution {
def reformatDate(date: String): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec reformat_date(date :: String.t) :: String.t
def reformat_date(date) do

end
end
```

Erlang:

```
-spec reformat_date(Date :: unicode:unicode_binary()) ->
unicode:unicode_binary().
reformat_date(Date) ->
.
```

Racket:

```
(define/contract (reformat-date date)
(-> string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Reformat Date
* Difficulty: Easy
* Tags: string
*
* 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 reformatDate(string date) {

}
};
```

Java Solution:

```
/**
* Problem: Reformat Date
* Difficulty: Easy
* Tags: string
*
* 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 reformatDate(String date) {

}
}
```

Python3 Solution:

```
"""
Problem: Reformat Date
Difficulty: Easy
Tags: string

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 reformatDate(self, date: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def reformatDate(self, date):
"""
:type date: str
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Reformat Date
* Difficulty: Easy
* Tags: string
*
* 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} date
* @return {string}
*/
var reformatDate = function(date) {

};
```

TypeScript Solution:

```
/**
* Problem: Reformat Date
* Difficulty: Easy
* Tags: string
*
* 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 reformatDate(date: string): string {

};
```

C# Solution:

```
/*
* Problem: Reformat Date
* Difficulty: Easy
* Tags: string
*
* 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 ReformatDate(string date) {

}
}
```

C Solution:

```
/*
* Problem: Reformat Date
* Difficulty: Easy
* Tags: string
*
* 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* reformatDate(char* date) {

}
```

Go Solution:

```
// Problem: Reformat Date
// Difficulty: Easy
// Tags: string
//
// 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 reformatDate(date string) string {

}
```

Kotlin Solution:

```
class Solution {
fun reformatDate(date: String): String {

}
}
```

Swift Solution:

```
class Solution {
func reformatDate(_ date: String) -> String {
}
}
```

Rust Solution:

```
// Problem: Reformat Date
// Difficulty: Easy
// Tags: string
//
// 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 reformat_date(date: String) -> String {

}
}
```

Ruby Solution:

```
# @param {String} date
# @return {String}
def reformat_date(date)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $date
* @return String
*/
function reformatDate($date) {

}
}
```

Dart Solution:

```
class Solution {
String reformatDate(String date) {

}
}
```

Scala Solution:

```
object Solution {
def reformatDate(date: String): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec reformat_date(date :: String.t) :: String.t
def reformat_date(date) do

end
end
```

Erlang Solution:

```
-spec reformat_date(Date :: unicode:unicode_binary()) ->
unicode:unicode_binary().
reformat_date(Date) ->
.
```

Racket Solution:

```
(define/contract (reformat-date date)
(-> string? string?)
)
```