Problem 1185: Day of the Week

Problem Information

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

Problem Description

Given a date, return the corresponding day of the week for that date.

The input is given as three integers representing the

day

,

month

and

year

respectively.

Return the answer as one of the following values

{"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"}

.

Note:

January 1, 1971 was a Friday.

Example 1:

Input:

day = 31, month = 8, year = 2019

Output:

"Saturday"

Example 2:

Input:

day = 18, month = 7, year = 1999

Output:

"Sunday"

Example 3:

Input:

day = 15, month = 8, year = 1993

Output:

"Sunday"

Constraints:

The given dates are valid dates between the years

1971

and

2100

.

Code Snippets

C++:

```
class Solution {
public:
string dayOfTheWeek(int day, int month, int year) {

}
};
```

Java:

```
class Solution {
public String dayOfTheWeek(int day, int month, int year) {

}
}
```

Python3:

```
class Solution:
def dayOfTheWeek(self, day: int, month: int, year: int) -> str:
```

Python:

```
class Solution(object):
def dayOfTheWeek(self, day, month, year):
"""
:type day: int
:type month: int
:type year: int
:rtype: str
"""
```

JavaScript:

```
/**
* @param {number} day
```

```
* @param {number} month
* @param {number} year
* @return {string}
*/
var dayOfTheWeek = function(day, month, year) {

};
```

TypeScript:

```
function dayOfTheWeek(day: number, month: number, year: number): string {

};
```

C#:

```
public class Solution {
public string DayOfTheWeek(int day, int month, int year) {

}
}
```

C:

```
char* dayOfTheWeek(int day, int month, int year) {

}
```

Go:

```
func dayOfTheWeek(day int, month int, year int) string {

}
```

Kotlin:

```
class Solution {
fun dayOfTheWeek(day: Int, month: Int, year: Int): String {

}
}
```

Swift:

```
class Solution {
func dayOfTheWeek(_ day: Int, _ month: Int, _ year: Int) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn day_of_the_week(day: i32, month: i32, year: i32) -> String {

}
}
```

Ruby:

```
# @param {Integer} day
# @param {Integer} month
# @param {Integer} year
# @return {String}
def day_of_the_week(day, month, year)

end
```

PHP:

```
class Solution {

/**
* @param Integer $day
* @param Integer $month
* @param Integer $year
* @return String
*/
function dayOfTheWeek($day, $month, $year) {

}
}
```

Dart:

```
class Solution {
String dayOfTheWeek(int day, int month, int year) {

}
}
```

Scala:

```
object Solution {
def dayOfTheWeek(day: Int, month: Int, year: Int): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec day_of_the_week(day :: integer, month :: integer, year :: integer) ::
String.t
def day_of_the_week(day, month, year) do

end
end
```

Erlang:

```
-spec day_of_the_week(Day :: integer(), Month :: integer(), Year ::
integer()) -> unicode:unicode_binary().
day_of_the_week(Day, Month, Year) ->
.
```

Racket:

```
(define/contract (day-of-the-week day month year)
(-> exact-integer? exact-integer? exact-integer? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Day of the Week
* Difficulty: Easy
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
string dayOfTheWeek(int day, int month, int year) {

}
};
```

Java Solution:

```
/**
* Problem: Day of the Week
* Difficulty: Easy
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public String dayOfTheWeek(int day, int month, int year) {

}
}
```

Python3 Solution:

```
"""
Problem: Day of the Week
Difficulty: Easy
Tags: math
```

```
Approach: Optimized algorithm based on problem constraints
Time Complexity: O(n) to O(n^2) depending on approach
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def dayOfTheWeek(self, day: int, month: int, year: int) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def dayOfTheWeek(self, day, month, year):
"""
:type day: int
:type month: int
:type year: int
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Day of the Week
* Difficulty: Easy
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number} day
* @param {number} month
* @param {number} year
* @return {string}
*/
var dayOfTheWeek = function(day, month, year) {
```

```
};
```

TypeScript Solution:

```
/**
* Problem: Day of the Week
* Difficulty: Easy
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

function dayOfTheWeek(day: number, month: number, year: number): string {

};
```

C# Solution:

```
/*
* Problem: Day of the Week
* Difficulty: Easy
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* 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 string DayOfTheWeek(int day, int month, int year) {

}
}
```

C Solution:

```
/*
* Problem: Day of the Week
* Difficulty: Easy
```

```
* Tags: math
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

char* dayOfTheWeek(int day, int month, int year) {

}
```

Go Solution:

```
// Problem: Day of the Week
// Difficulty: Easy
// Tags: math
//
// Approach: Optimized algorithm based on problem constraints
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(1) to O(n) depending on approach

func dayOfTheWeek(day int, month int, year int) string {

}
```

Kotlin Solution:

```
class Solution {
fun dayOfTheWeek(day: Int, month: Int, year: Int): String {

}
}
```

Swift Solution:

```
class Solution {
func dayOfTheWeek(_ day: Int, _ month: Int, _ year: Int) -> String {

}
}
```

Rust Solution:

```
// Problem: Day of the Week
// Difficulty: Easy
// Tags: math
//
// Approach: Optimized algorithm based on problem constraints
// 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 day_of_the_week(day: i32, month: i32, year: i32) -> String {

}
}
```

Ruby Solution:

```
# @param {Integer} day
# @param {Integer} month
# @param {Integer} year
# @return {String}
def day_of_the_week(day, month, year)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $day
* @param Integer $month
* @param Integer $year
* @return String
*/
function dayOfTheWeek($day, $month, $year) {

}
}
```

Dart Solution:

```
class Solution {
String dayOfTheWeek(int day, int month, int year) {

}
}
```

Scala Solution:

```
object Solution {
def dayOfTheWeek(day: Int, month: Int, year: Int): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec day_of_the_week(day :: integer, month :: integer, year :: integer) ::
String.t
def day_of_the_week(day, month, year) do

end
end
```

Erlang Solution:

```
-spec day_of_the_week(Day :: integer(), Month :: integer(), Year ::
integer()) -> unicode:unicode_binary().
day_of_the_week(Day, Month, Year) ->
.
```

Racket Solution:

```
(define/contract (day-of-the-week day month year)
(-> exact-integer? exact-integer? exact-integer? string?)
)
```