SOFE3980U

Homework Assignment

NextDate-Test case design

Ashwin Prem

2024

<https://github.com/ashwinprem/nextdatehomeworkk>

The NextDate functions has three inputs: Day, Month, Year

Characteristics:

* Day
* Month
* Year

Domain:

* Day (1-31)
* Month (1-12)
* Year (1812 – 2212)

Blocks:

* Normal value – Entering a date will output the next day
* Max value – Entering a date that is the last value of the domain would output a date that resets to the minimum value of the domain
* Invalid value – Entering a date outside of the domain would yield an error
* Leap Year value – Takes into consideration leap years, such that February’s domain is increased by one day.
* Leap century value – years are that divisible by 100 but not by 400 are not leap years

Table 1: Testing critical dates that would validate the normal functionality of the method/function

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case** | **Day** | **Month** | **Year** | **Expected value**  **(DD/MM/YYYY)** | **Notes** |
| 1 | 1 | 1 | 1912 | 02/01/1912 | Results next day |
| 2 | 31 | 1 | 1912 | 01/02/1912 | Day loops back to 1. Month increments. |
| 3 | 28 | 2 | 1913 | 01/03/1913 | Non-leap-year. 28 is the last day of February |
| 4 | 31 | 12 | 1912 | 01/01/1913 | Day and Month loops back to 1. Year increments |

Table 2: Tests most out of bounds inputs for all domains.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case** | **Day** | **Month** | **Year** | **Expected value**  **(DD/MM/YYYY)** | **Notes** |
| 5 | 1 | 1 | 1711 | (--/--/----) | Invalid, out of year’s domain |
| 6 | 1 | 1 | 2213 | (--/--/----) | Invalid, out of year’s domain |
| 7 | 0 | 1 | 1912 | (--/--/----) | Invalid, out of day’s domain |
| 8 | 32 | 1 | 1912 | (--/--/----) | Invalid, out of day’s domain |
| 9 | 1 | 0 | 1912 | (--/--/----) | Invalid, out of month’s domain |
| 10 | 1 | 13 | 1912 | (--/--/----) | Invalid, out of month’s domain |

Table 3: Testing if day 31 constitutes as invalid entry for months that don't contain 31 days.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case** | **Day** | **Month** | **Year** | **Expected value**  **(DD/MM/YYYY)** | **Notes** |
| 11 | 30 | 2 | 1912 | (--/--/----) | Invalid, day out of bound for month of feb |
| 12 | 31 | 4 | 1912 | (--/--/----) | Invalid, day out of bound for month of april |
| 13 | 31 | 6 | 1912 | (--/--/----) | Invalid, day out of bound for month of June |
| 14 | 31 | 8 | 1912 | (--/--/----) | Invalid, day out of bound for month of August |
| 15 | 31 | 9 | 1912 | (--/--/----) | Invalid, day out of bound for month of September |
| 16 | 31 | 11 | 1912 | (--/--/----) | Invalid, day out of bound for month of November |

Table 4: Tests to validate the correct functionality of leap years and leap centuries

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case** | **Day** | **Month** | **Year** | **Expected value**  **(DD/MM/YYYY)** | **Notes** |
| 17 | 28 | 2 | 2024 | (29/02/2024) | Checking if years divisible by 4 are correctly leap years. |
| 18 | 29 | 2 | 1912 | (01/03/1912) | Checking if Day 29 for February is accepted as valid for leap years, and successfully increments |
| 19 | 28 | 2 | 1900 | (01/03/1900) | Checking if centuries divisible by 100 but not 400 correctly aren’t leap years. |
| 20 | 28 | 2 | 2000 | (29/02/2000) | Checking if centuries divisible by both 100 and 400 correctly are leap years. |

A screenshot of a computer error message

Description automatically generated

Figure 1: Successfully built and ran the tests.

A screenshot of a computer

Description automatically generated

A computer screen shot of a program code

Description automatically generated A computer code with text

Description automatically generated with medium confidence