Ad-Soyad: Alper YALMAN 12.08.2023

DataComputation Functions Test Cases

Girilen tarihe göre önceki ve sonraki günün tarihlerini hesaplayan DataComputation programında aşağıda belirtilen 4 adet fonksiyon yazılmıştır.

1. bool IsLeapYear(int year)

IsLeapYear fonksiyonu, girilen tarihe ait yılın artık yıl olup olmadığını hesaplamaktadır. Girilen yılın artık yıl olup olmadığı bilgisi, sonraki fonksiyonlarda ilgili yıla ait Şubat ayının 28 veya 29 gün olmasını belirlemek üzere kullanılacaktır. Artık yıl koşulu aşağıda verilmektedir:

Condition: *((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0)*

Bu koşulu dikkate alarak IsLeapYear fonksiyonunu test etmek için aşağıdaki durumlar göz önünde bulundurulmalıdır.

NOT: IsLeapYear fonksiyonunun tarih geçerlilik kontrolü yapılan IsValidYear fonksiyonundan sonra çağrıldığı varsayılarak, yılın negatiflik kontrolü vb. girdiler burada verilmemiştir. IsLeapYear fonksiyonuna geçerlilik kontrolü yapılmış bir yıl verisinin geldiği varsayılmaktadır.

Case 1: *400'ün katı olan yıllar*

400'ün katı olan tüm yıllar artık yıl olduğundan fonksiyon 400'ün katı olarak verilen tüm yıllar için "***true"*** değerini döndürmelidir. 400'e bölünebilme koşulu karmaşık bir işlem olmadığından birkaç adet girdi ile kontrol etmenin yeterli olacağı düşünülmektedir. Aşağıda 400'ün katı olan 5 adet girdi belirtilmiştir. Fonksiyonun tüm girdiler için "***true"*** değerini döndürmesi beklenmektedir.

Inputs	Expected Outputs
800	True
1200	True
1600	True
2000	True
2400	True

Case 2: *400'ün katı olmayan yıllar*

400'ün katı olmayan yıllar için verilen yılın 4'e ve 100'e bölünme koşulları göz önünde bulundurulmalıdır.

Case 2.1: *4'ün katı ve 100'ün katı olmayan yıllar*

Artık yılı sağlayan diğer koşuldur. Bu durumu sağlayan girdiler için fonksiyonun "***true"*** değerini döndürmesi beklenmektedir. Aşağıda 4'ün katı ve 100'ün katı olmayan 5 adet girdi belirtilmiştir.

Inputs	Expected Outputs
1248	True
1680	True
1960	True
2004	True
2024	True

Case 2.2: *4'ün ve 100'ün katı olan yıllar*

4'ün katı olup aynı zamanda 100'ün de katı olan yıllar (400'ün katı olmamak koşuluyla), artık yıl koşulunu sağlamamaktadır. Bu durumu oluşturan girdiler için fonksiyonun "***false***" değerini döndürmesi beklenmektedir. Aşağıda 4'ün ve 100'ün katı olan (ancak 400'ün katı olmayan) 5 adet girdi belirtilmiştir.

Inputs	**Expected Outputs**
900	False
1300	False
1700	False
2100	False
2300	False

Case 2.3: *4'ün ve 100'ün katı olmayan yıllar*

4'ün ve 100'ün de katı olmayan yıllar artık yıl koşulunu sağlamamaktadır. Bu durumu oluşturan girdiler için fonksiyonun "***false***" değerini döndürmesi beklenmektedir. Aşağıda 4'ün ve 100'ün katı olmayan 10 adet girdi belirtilmiştir.

Inputs	**Expected Outputs**
1302	False
1709	False
1881	False
1905	False
1950	False
1975	False
1990	False
1997	False
2001	False
2006	False

Sonuç olarak IsLeapYear fonksiyonu için toplamda **4 farklı durum** göz önünde bulunudurulmuş ve **25 adet farklı girdiye** karşılık elde edilmesi beklenen çıktılar belirlenmiştir. Programda tanımlanabilecek limitler dahilinde girdi sayıları ve koşulları değişkenlik gösterebilecektir.

2. bool IsValidDate(struct CDate givenDate)

IsValidDate fonksiyonu, girdi olarak aldığı CDate tipindeki tarih verisinin geçerli bir tarih olup olmadığını kontrol etmektedir. CDate veri yapısı aşağıda verildiği gibidir.

```
struct CDate {
        int mYear;
        int mMonth;
        int mDay;
};
```

Fonksiyon, geçerli olan tarihler için "true", geçersiz olan tarihler için "false" değerini döndürecektir. Bu fonksiyonda aşağıdaki durumlar göz önünde bulundurulmalıdır.

Ad-Soyad: Alper YALMAN 12.08.2023

Case 1: *givenDate.mYear değerinin programda tanımlanan minYear değerinden küçük olup olmama durumu*

Programda bir minYear değişkeni kullanılmaktadır. Kullanıcının minYear değerinden daha küçük bir değer **girmemesi** beklenmektedir, aksi durumda fonksiyon "false" değerini döndürecektir. minYear değerinden büyük veya eşit bir değer girilmesi durumunda fonksiyon "true" değerini döndürecektir. Bu işlem için ay ve gün değerlerinden bağımsız olarak sadece yıl için sınır koşullarının değerlendirilmesi yeterli olacaktır.

Inputs	Expected Outputs
{minYear-1, MM, DD}	False
{minYear, MM, DD}	True
{minYear+1, MM, DD}	True

Case 2: *givenDate.mMonth değerinin [1, 12] aralığında olup olmaması durumu*

Yıl koşulunun sağlandığı varsayılarak; yılda 12 ay olduğundan kullanıcından ay değerinin [1, 12] aralığında girmesi beklenmektedir. Sınır değerlerde dahil olmak üzere bu aralıkta verilen değerler için fonksiyon "true", bu aralık dışındaki değerler için ise fonksiyon "false" değerini döndürecektir. Burada sınır değerleriyle birlikte, belirtilen aralıktan 1-2 değer seçilip test edilmesi yeterli görünmektedir. Gün değeri sonraki adımda kontrol edileceğinden bu durum için sabit olarak seçilebilir.

Inputs	Expected Outputs
{YYYY, 0, DD}	False
{YYYY, 1, DD}	True
{YYYY, 3, DD}	True
{YYYY, 9, DD}	True
{YYYY, 12, DD}	True
{YYYY, 13, DD}	False

Case 3: *givenDate.mDay değerinin [1, ilgili ayın maksimum gün sayısı] aralığında olup olmaması durumu*

Yıl ve ay koşullarının sağlandığı varsayılarak; gün değerinin minimum ve girilen ayın maksimum değeri aralığında olup olmadığı kontrol edilmektedir. Her bir ayın maksimum gün sayısı program içerisinde bir dizide tutulmaktadır. Burada Şubat ayı için bir istisna söz konusudur. Daha önceki artık yıl hesaplamasının sonucuna bağlı olarak Şubat ayı için maksimum gün sayısı 28 veya 29 olarak belirlenmektedir. Bu sebeple girdilerde artık yıl olup olmaması da önem arz etmektedir.

Inputs	Expected Outputs
{YYYY, MM, 0}	False
{YYYY, MM, 1}	True
{YYYY, 1, 15}	True
{YYYY, 1, 31}	True
{YYYY, 1, 32}	False
{2000, 2, 15}	True
{2000, 2, 29}	True
{2001, 2, 30}	False
{2001, 2, 15}	True
{2001, 2, 28}	True

{2001, 2, 29}	False
{YYYY, 3, 15}	True
{YYYY, 3, 31}	True
{YYYY, 3, 32}	False
{YYYY, 4, 15}	True
{YYYY, 4, 30}	True
{YYYY, 4, 31}	False
{YYYY, 5, 15}	True
{YYYY, 5, 31}	True
{YYYY, 5, 32}	False
{YYYY, 6, 15}	True
{YYYY, 6, 30}	True
{YYYY, 6, 31}	False
{YYYY, 7, 15}	True
{YYYY, 7, 31}	True
{YYYY, 7, 32}	False
{YYYY, 8, 15}	True
{YYYY, 8, 31}	True
{YYYY, 8, 32}	False
{YYYY, 9, 15}	True
{YYYY, 9, 30}	True
{YYYY, 9, 31}	False
{YYYY, 10, 15}	True
{YYYY, 10, 31}	True
{YYYY, 10, 32}	False
{YYYY, 11, 15}	True
{YYYY, 11, 30}	True
{YYYY, 11, 31}	False
{YYYY, 12, 15}	True
{YYYY, 12, 31}	True
{YYYY, 12, 32}	False

İlk iki girdi için sadece minimum gün değerindeki sınır koşulu test edildiğinden yıl ve ay sabit seçilebilir, ancak istenirse her bir ay için de bu kontrol yapılabilir. Yukarıdaki girdilerde sabit bir ay için kontrol edilmiştir. Sonraki koşullarda sırayla ilgili her bir ay için ay ortasından sabit bir gün, ilgili ayın maksimum değeri ve maksimumdan bir sonraki değeri verilerek fonksiyon test edilmektedir. Şubat ayı için artık yıl olan ve olmayan iki farklı yıl seçilerek girdi sağlanmıştır. Şubat dışındaki diğer aylarda yıl sabit olarak seçilebilir.

Sonuç olarak IsValidDate fonksiyonu için toplamda **3 farklı durum** göz önünde bulundurulmuş ve **50 adet farklı girdiye** karşılık elde edilmesi beklenen çıktılar belirlenmiştir. Programda tanımlanabilecek limitler dahilinde girdi sayıları ve koşulları değişkenlik gösterebilecektir.

3. struct CDate ComputeNextDate(struct CDate givenDate)

ComputeNextDate fonksiyonu, girdi olarak aldığı *givenDate* verisini kullanarak bir sonraki günün tarihini hesaplamaktadır. Bu fonksiyon çağırılmada önce IsValidDate fonksiyonu ile girilen tarihin kontrol edildiği göz önünde bulundurularak, givenDate verisinin geçerli bir tarihe karşılık geldiği varsayılmaktadır.

Ad-Soyad: Alper YALMAN 12.08.2023

Case 1: *Girilen tarihin yılın son günü olma durumu*

Girilen tarih yılın son günü ise, fonksiyonun çıktısı bir sonraki yılın ilk ayının ilk gününe karşılık gelmelidir. Yıl değeri sabit bir değer olarak seçilebilir.

Inputs	Expected Outputs
{YYYY, 12, 31}	{YYYY+1, 1, 1}

Case 2: *Girilen tarihin ilgili ayın son günü olma durumu (Yılın son günü hariç)*

Girilen tarihteki gün ilgili ayın son günü ise, fonksiyonun çıktısı bir sonraki ayın ilk gününe karşılık gelmelidir. Yıl değeri sabit bir değer olarak seçilebilir. Ancak artık yıllarda Şubat ayının son gününün değeri değişkenlik gösterdiğinden, Şubat ayı için artık yıl olup olmama durumu da göz önünde bulundurulmalıdır.

Inputs	Expected Outputs
{YYYY, 1, 31}	{YYYY, 2, 1}
{2000, 2, 29}	{2000, 3, 1}
{2001, 2, 28}	{2001, 3, 1}
{YYYY, 3, 31}	{YYYY, 4, 1}
{YYYY, 4, 30}	{YYYY, 5, 1}
{YYYY, 5, 31}	{YYYY, 6, 1}
{YYYY, 6, 30}	{YYYY, 7, 1}
{YYYY, 7, 31}	{YYYY, 8, 1}
{YYYY, 8, 31}	{YYYY, 9, 1}
{YYYY, 9, 30}	{YYYY, 10, 1}
{YYYY, 10, 31}	{YYYY, 11, 1}
{YYYY, 11, 30}	{YYYY, 12, 1}

Case 3: *Girilen tarihin ayın son günü* ***olmama*** *durumu*

Case 1 ve Case 2'deki özel durumların dışında girilen herhangi bir tarih için fonksiyonun çıktısı aynı ay ve yılın bir sonraki gününe karşılık gelmelidir. Yıl değeri sabit bir değer olarak seçilebilir. Şubat ayı için artık yıl olup olmama durumu da göz önünde bulundurulmalıdır.

Inputs	Expected Outputs
{YYYY, 1, 14}	{YYYY, 1, 15}
{2000, 2, 28}	{2000, 2, 29}
{2001, 2, 27}	{2001, 2, 28}
{YYYY, 3, 15}	{YYYY, 3, 16}
{YYYY, 4, 16}	{YYYY, 4, 17}
{YYYY, 5, 17}	{YYYY, 5, 18}
{YYYY, 6, 18}	{YYYY, 6, 19}
{YYYY, 7, 19}	{YYYY, 7, 20}
{YYYY, 8, 20}	{YYYY, 8, 21}
{YYYY, 9, 21}	{YYYY, 9, 22}
{YYYY, 10, 22}	{YYYY, 10, 23}
{YYYY, 11, 23}	{YYYY, 11, 24}
{YYYY, 12, 24}	{YYYY, 12, 25}

Sonuç olarak ***ComputeNextDate*** fonksiyonu için toplamda **3 farklı durum** göz önünde bulunudurulmuş ve **26 adet farklı girdiye** karşılık elde edilmesi beklenen çıktılar belirlenmiştir. Programda tanımlanabilecek limitler dahilinde girdi sayıları ve koşulları değişkenlik gösterebilecektir.

4. struct CDate ComputePreviousDate(struct CDate givenDate)

ComputePreviousDate fonksiyonu, girdi olarak aldığı *givenDate* verisini kullanarak bir önceki günün tarihini hesaplamaktadır. Bu fonksiyon çağırılmada önce IsValidDate fonksiyonu ile girilen tarihin kontrol edildiği göz önünde bulundurularak, givenDate verisinin geçerli bir tarihe karşılık geldiği varsayılmaktadır.

Case 1: *Girilen tarihin yılın ilk günü olma durumu*

Girilen tarih yılın ilk günü ise, fonksiyonun çıktısı bir önceki yılın son ayının son gününe karşılık gelmelidir. Yıl değeri sabit bir değer olarak seçilebilir.

Inputs	Expected Outputs
{YYYY, 1, 1}	{YYYY-1, 12, 31}

Case 2: *Girilen tarihin ilgili ayın ilk günü olma durumu (yılın ilk günü hariç)*

Girilen tarihteki gün ilgili ayın ilk günü ise, fonksiyonun çıktısı aynı yılın bir önceki ayının son gününe karşılık gelmelidir. Yıl değeri sabit bir değer olarak seçilebilir. Ancak artık yıllarda Şubat ayının son gününün değeri değişkenlik gösterdiğinden, Şubat ayı için artık yıl olup olmama durumu da göz önünde bulundurulmalıdır.

Inputs	Expected Outputs
{YYYY, 2, 1}	{YYYY, 1, 31}
{2000, 3, 1}	{2000, 2, 29}
{2001, 3, 1}	{2001, 2, 28}
{YYYY, 4, 1}	{YYYY, 3, 31}
{YYYY, 5, 1}	{YYYY, 4, 30}
{YYYY, 6, 1}	{YYYY, 5, 31}
{YYYY, 7, 1}	{YYYY, 6, 30}
{YYYY, 8, 1}	{YYYY, 7, 31}
{YYYY, 9, 1}	{YYYY, 8, 31}
{YYYY, 10, 1}	{YYYY, 9, 30}
{YYYY, 11, 1}	{YYYY, 10, 31}
{YYYY, 12, 1}	{YYYY, 11, 30}

Case 3: *Girilen tarihin ayın ilk günü* ***olmama*** *durumu*

Case 1 ve Case 2'deki özel durumların dışında girilen herhangi bir tarih için fonksiyonun çıktısı aynı ay ve yılın bir önceki gününe karşılık gelmelidir. Yıl değeri sabit bir değer olarak seçilebilir. Şubat ayı için artık yıl olup olmama durumu da göz önünde bulundurulmalıdır.

Inputs	Expected Outputs
{YYYY, 1, 14}	{YYYY, 1, 13}
{2000, 2, 29}	{2000, 2, 28}
{2001, 2, 28}	{2001, 2, 27}
{YYYY, 3, 15}	{YYYY, 3, 14}
{YYYY, 4, 16}	{YYYY, 4, 15}
{YYYY, 5, 17}	{YYYY, 5, 16}

{YYYY, 6, 18}	{YYYY, 6, 17}
{YYYY, 7, 19}	{YYYY, 7, 18}
{YYYY, 8, 20}	{YYYY, 8, 19}
{YYYY, 9, 21}	{YYYY, 9, 20}
{YYYY, 10, 22}	{YYYY, 10, 21}
{YYYY, 11, 23}	{YYYY, 11, 22}
{YYYY, 12, 24}	{YYYY, 12, 23}

Sonuç olarak ***ComputePreviousDate*** fonksiyonu için toplamda **3 farklı durum** göz önünde bulunudurulmuş ve **26 adet farklı girdiye** karşılık elde edilmesi beklenen çıktılar belirlenmiştir. Programda tanımlanabilecek limitler dahilinde girdi sayıları ve koşulları değişkenlik gösterebilecektir.