Assignment 4: Easter

Course 'Imperative Programming' (IPC031)

Make sure to start working on the assignment **before the lab** session,
otherwise you will be too late for the deadline

1 Background

In this assignment you structure your programs by means of parameterized functions that may return values. This assignment starts with the use of unit testing. On Brightspace you can find documentation on unit testing, including a .zip file with a test assignment to test your setup. More on the test assignment is explained in the documentation on Brightspace.

2 Learning objectives

After doing this assignment you are able to:

- analyze the operational behavior of a program using parameterized functions.
- structure your program by means of value-returning, parameterized functions.
- establish a level of confidence in the quality of your code by means of unit testing.

3 Assignment

In this assignment you design and implement a program that computes the dates of a number of christian holy days that depend on the date of easter. You find a file "`assignment-04-mandatory-files.zip`" on Brightspace that contains a "`main.cpp`" that you can use to create the functions of Part 1 and Part 2. In addition, the "`main_test.cpp`" contains all the commented unit tests for these functions. After developing a function, first proceed with unit testing that function by removing the comment surrounding the unit test of that particular function. Fix any issues before continuing with the remainder of the assignment. For any auxiliary function that you develop, add at least one non-trivial unit test to "`main_test.cpp`" and test that function.

Part 1: Leap years

When calculating with dates, you need to take leap years into account. A year is a leap year if it is divisible by 4, except if it is a multiple of 100 that is not divisible by 400. For example, 1600 is a leap year, but 1700 is not a leap year. In a leap year, February has 29 days instead of 28. A complete list of months and number of days is:

Month	**No of days**	**Month**	**No of days**	**Month**	**No of days**
January	31	May	31	September	30
February	28 (29 in a leap year)	June	30	October	31
March	31	July	31	November	30
April	30	August	31	December	31

Constraints:

- Design and implement a parameterized function:

```
bool is_leap_year (int year)
```

  This function returns `true` only if `year` is a leap year, and returns `false` otherwise.
- Design and implement the parameterized function:

```
int number_of_days_in_month (int year, Month month)
```

  This function returns the number of days of month in year, by taking leap years into account as well. `Month` is already given as an enumeration type in "`main.cpp:`"

```
enum Month { January = 1,February, March, April, June, July, August, September, October, November, December };
```

Part 2: Holy days based on easter

The dates of several christian holy days depend on easter in the following way:

- Carnival: 7 weeks before easter, ending on Tuesday.
- Good Friday: Friday before easter.
- Ascension Day: 10 days before whitsuntide.
- Whitsuntide: 7 weeks after easter.

Easter is on the first Sunday after the first full moon on or after the beginning of spring (March 21). The Meeus/Jones/Butcher formula computes the easter date (month, date) for a year (Y):

$$
\begin{aligned}
a &= Y \bmod 19 \\
b &= Y/100 \\
c &= Y \bmod 100 \\
d &= b/4 \\
e &= b \bmod 4 \\
f &= (b+8)/25 \\
g &= (b-f+1)/3 \\
h &= (19 \times a + b - d - g + 15) \bmod 30 \\
i &= c/4 \\
k &= c \bmod 4 \\
L &= (32 + 2 \times e + 2 \times i - h - k) \bmod 7 \\
m &= (a + 11 \times h + 22 \times L)/451 \\
\text{month} &= (h + L - 7 \times m + 114)/31 \\
\text{day} &= ((h + L - 7 \times m + 114) \bmod 31) + 1
\end{aligned}
$$

It is important to observe that all computations are integer operations. You can find the correct C++-version of this formula in “`main.cpp`” (the functions `easter_base`, `calculate_easter_day`, and `calculate_easter_month`).

- Design and implement functions:

```
string show_carnival      (int easter_day, Month easter_month, int year)
string show_easter        (int easter_day, Month easter_month)
string show_good_friday   (int easter_day, Month easter_month, int year)
string show_whitsuntide   (int easter_day, Month easter_month, int year)
string show_ascension_day (int easter_day, Month easter_month, int year)
```

  These functions should return the string “day-month” for each holy day.
- Design and implement a function

```
void show_holy_days (int year)
```

  that allows a user to enter a year Y and then prints the dates “day-month” of all of the above mentioned holy days (including easter) for that year Y.
- Design and implement an intelligible user interface: for the user of your application it should be clear what input is expected by the program and how the output can be interpreted.

The `string` type will be explained in detail in week 7. For this assignment, it suffices to use the following functions to work with `string` values:

- Declaring an empty `string` variable: `string text = "";`
- Appending a `string` value: `text = text + "Hello";`
- Appending a `char` value: `text = text + '␣';`
- Appending an `int` value: `text = text + to_string (42);`
- Appending the value of a `string` variable `t`: `text = text + t;`
- Appending the value of a `char` variable `c`: `text = text + c;`
- Appending the value of an `int` variable `n`: `text = text + to_string (n);`

4 Products

As product-to-deliver you upload to Brightspace "`main.cpp`" that you have created with solutions for each part of the assignment and "`main_test.cpp`" that has been extended with non-trivial unit tests for each auxiliary function that you have developed in "`main.cpp`".

An important reminder from the lecture: **If your implementation fails a provided test you cannot obtain grade 'Good'.**

Deadline

Lab assignment: Friday, September 29, 2023, 23:59h