Assignment week 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.

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. After developing a function, first proceed with unit testing that function by adding tests to the functions `run_tests_part_one` and `run_tests_part_two`. Fix any issues before continuing with the remainder of the assignment.

Important

You are not allowed to change the signature of functions defined in “`main.cpp`”, unless instructed otherwise. Specifically this means you may not:

- Rename the function
- Add or remove parameters
- Change the type of parameters
- Change the order of parameters
- Change the return type

You may of course still introduce new auxiliary functions with a signature of your own choosing, as long as you use them to implement the exact required functions. This restriction allows us to properly test your code while grading. Violating it may thus prevent us from properly grading your work, resulting in a “Fail”. This restriction applies to all remaining assignments, both bonus and mandatory, but will not be repeated.

Part 1: Leap years

Part 1.1: Programming

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 1.2: Testing

Extend the function `run_tests_part_one` with at least 2 input/output tests for all of the functions you developed above. Of course, you might (and are encouraged to) add more than 2 tests per function. Make sure that you obtain good test coverage.
The provided `main` function in the template will run the tests in the function `run_tests_part_one`.

Part 2: Holy days based on easter

Part 2.1: Programming

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):

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

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. Do not output leading zeros, *i.e.*, March 6th should return “6-4” and not “06-04”.
- Design and implement a function

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

  that given a year Y, prints the dates “day-month” of all of the above mentioned holy days (including easter) for that year Y.
- In the `main` function (below the code that runs the tests), design and implement an intelligible user interface. Users should be able to enter a year, and then see the corresponding holy days. 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);`

Part 2.2: Testing

Extend the function `run_tests_part_two` with at least 2 input/output tests for all of the functions you developed above. Of course, you might (and are encouraged to) add more than 2 tests per function. Make sure that you obtain good test coverage.
The provided `main` function in the template will run the tests in the function `run_tests_part_two`.

4 Products

As product-to-deliver you upload to Brightspace:

- “`main.cpp`” that you have created with solutions for each part of the assignment.
- To obtain the grade ‘Sufficient’, you should have at least 2 input/output tests for each function you develop.
- To obtain the grade ‘Good’, your tests should have good coverage.

Deadline

Lab assignment: Friday, September 27, 2024, 23:59h

Important notes:

1. check that you have actually submitted your solution in Brightspace.
2. the deadline is firm, and it is impossible to upload a solution after expiry. In that case you fail the assignment.
3. you can upload solutions to Brightspace several times, but only the last submission is stored.
4. identify yourself and your lab partner in every uploaded document. The identification consists of your first and last names, student numbers, and number of (sub) assignment. By identifying yourself, you declare that the uploaded work has been created solely by you and your lab partner.
5. your work will be judged and commented upon. We expect that you obtain the feedback, read it, and use it to for the next exercises.

6. it is essential that you only submit your own solution, never copy somebody/something else’s solution, and never share your solution—in particular: **AI tools (including but not limited to Github Copilot or ChatGPT) are not permitted**, solutions from previous year cannot be reused, and finally, you and your lab partner take joint responsibility for the assignment you submit.