

C++: Practical session 7

1 Date class

Throughout the lectures, you have seen various skeleton/pseudocode versions of a `Date` class. Your task now is to implement such a class to a reasonable level of completeness.

In programming, it is often useful to create a series of checks (called unit-tests) that ensure a function or class behaves as it should, particularly when used on unusual data, or at least data that you (as the programmer) know requires coding effort to handle correctly.

When finished, the following code (or something very like it) should work, using your `Date` class:

```
#include <cassert>

int main(void){
    Date startOfTerm(6, 10, 2023);
    startOfTerm.advance();

    assert( startOfTerm.day() == 7 );
    assert( startOfTerm.month() == 10 );
    assert( startOfTerm.year() == 2023 );

    Date halloween(31, 10, 2023);
    halloween.advance();

    assert( halloween.day() == 1 );
    assert( halloween.month() == 11 );
    assert( halloween.year() == 2023 );

    /* This should fail (exit) at run-time because
       there is no 31st November. */
    halloween.setDay(31);

    halloween.set(31, 10, 2023); // This should succeed

    /* This should fail (exit) at run-time because
       there is no 31st November. */
    halloween.setMonth(11);

    Date startOfYear(1, 1, 2024);
    startOfYear.back();
    assert( startOfYear.day() == 31 );
    assert( startOfYear.month() == 12 );
    assert( startOfYear.year() == 2023 );

    const Date endOfYear = startOfYear;
    assert(endOfYear.day() == 31);
    endOfYear.setMonth(10); // Should fail at *compile-time*
    endOfYear = halloween; // Should fail at *compile-time*
}
```

2 Extensions

1. Make sure that your `Date` class deals with leap-years correctly. Create your own tests that the class should pass, following the above format.
2. Now look up the following, and related, C functions: `strptime`, `localtime`, and the structs `tm` and `timeval`.
3. Never try to create your own `Date` class for real-world applications; always use a library. Dates are hard.