Date

Submission deadline:	**2022-04-10 23:59:59**
Late submission with malus:	**2022-05-15 23:59:59** (Late submission malus: 100.0000 %)
Evaluation:	**5.5000**
Max. assessment:	**5.0000** (Without bonus points)
Submissions:	2 / 25 Free retries + 20 Penalized retries (-2 % penalty each retry)
Advices:	1 / 2 Advices for free + 2 Advices with a penalty (-10 % penalty each advice)

The task is to develop class `CDate` to represent a date. The dates will follow Gregorian calendar rules. It is required that at least years 2000 to 2030 are handled by the class.

The class shall have the following interface:

- constructor with parameters (y,m,d) initializes an instance of the class with the date given. The constructor shall test the date. If the date is invalid, `InvalidDateException` shall be thrown,
- operator + can be used to add an integer to an instance of `CDate`. The operation returns a date which is shifted to the future by the given number of days (if the integer is negative, it shifts to the past),
- operator - can be used to subtract an integer from `CDate`. The result is a date shifted by the given number of days to the past (to the future if the integer is negative),
- operator - can be used to subtract two instances of `CDate`, the result is an integer equal to the number of days between the two dates,
- operator ++ and -- in bot prefix and postfix notation can be used to increase/decrease the date by one day. The operators shall follow the usual behavior,
- operators ==, !=, <, <=, >, and >= can be used to compare two instances of `CDate`. Dates in the future are considered greater than dates in the past.
- operator << can be used to display the date in an output stream. The conversion shall use the ISO date format (%Y-%m-%d, e.g. 2000-01-31). The mandatory tests will use the ISO format only. The format may be modified by `date_format` manipulator, the manipulator must be implemented to pass the bonus test (it is not used in the mandatory tests),
- operator >> can be used to read date from an input stream. The mandatory tests assume default ISO date format. If the reading fails (invalid format, invalid date, ...), the operator sets fail bit in the corresponding stream and leaves the original contents of the `CDate` instance unmodified. Similarly to the output operator, bonus tests require an implementation of `date_format` manipulator to handle arbitrary input format.

Submit a source file with your `CDate` implementation. The class must follow the interface described above. If there is a mismatch in the interface, the compilation will fail. You may extend the interface and add you auxiliary methods and member variables (both public and private, although private are preferred). The submitted file must include both declarations as well as implementation of the classes (the methods may be implemented inline but do not have to). The submitted file shall not contain `main`, your tests, or any #include definitions. If present, please, keep them in a conditional compile block. Use the template as a basis for your implementation. If the preprocessor definitions are preserved, the file may be submitted to Progtest.

This homework does not aim at speed. Moreover, the range of required years is very limited. Thus standard library date functions may be used to convert/compare dates (however, the standard functions must be used with certain caution).

There are mandatory and bonus tests in the homework. The mandatory tests do not use the `date_format` manipulator, the default ISO format %Y-%m-%d is used for all conversions. The bonus test requires an implementation of the manipulator such that the date format may be set arbitrarily. If you decide not to implement the bonus, please, keep the dummy implementation provided in the sample. If the manipulator is not present at all, the program will not compile.

`date_format` manipulator uses the following formats:

- characters except percent character: if present in the input format, it means that the input shall contain that character. The character is to be skipped in the input stream. If present in the output format, it means that the character is to be copied to the output. An example is the dash character in the ISO format %Y-%m-%d,
- percent character is used to start day (%d), month (%m), or year (%Y) conversion. If present in the input format then the corresponding number (day, month, or year) is to be parsed from the input stream. If present in the output format, the corresponding day/month/year value shall be appended to the output stream (left padded by zero, 2 digits day+month/4 digits year),
- percent character followed by a character other than Y, m, and d (e.g. %x, %%, ...) represents the second character (i.e. x and % in the example). The "escaped" character is processed as in the first case.

The `date_format` manipulator is just like other manipulators (`hex`, `oct`, ...) in that it sets the conversion of all subsequent `CDate` processing in the stream until the format is modified by another `date_format` use. The manipulator does apply to the stream where it was used, it must not modify the behavior of other streams.

There are not any restrictions on output format (an extreme case is the "hello kitty" format below). The input format is restricted to guarantee unambiguous processing: each of %Y, %m and %d must be used exactly once in the format. The example shows invalid input formats (some conversion is missing/duplicate), these input conversions lead to an error in the subsequent reading.

Advice:

- A source with runs are available in the attached archive. The source contains mandatory tests as well as examples of the bonus test (with the manipulator).
- It may be surprising for you, however, some days are not 24 hours. This fact may be important for certain implementations.
- If implementing the bonus part, read C++ manual, methods `xalloc`, `register_callback`, `pword`, and `iword` in `ios_base` class.
- Your date reading operator should check the width of months and dates. Both values should be 2 digits wide, padded with an extra zero if needed. This check may be omitted in the mandatory tests (the values generated for these tests are correctly padded). On the other hand, the check must be thorough to pass the bonus test (if the check is not present, the reading with arbitrary date format may be ambiguous). There are some extra test cases included in the attached bonus tests.
- Based on your implementation, you may need to explicitly handle leap years. The basic rule is: a year divisible by 4 is a leap year (e.g., 2020, 2024, ...). There is an exception to this rule: if a year is divisible by 100, then it is **not** a leap year (e.g., years 1900 and 2100 were not/will not be leap years). Moreover, there is an exception of the exception: if a year is divisible by 400, then it is a leap year (e.g., year 2000 was a leap year). The homework is, however, simplified since the years are limited to the range 2000 - 2030.
- The solution of this homework is quite simple and straightforward. Therefore, solutions of this homework **cannot** be used for code review. On the other hand, a solution that correctly passes all tests (i.e., including bonus tests) is not that trivial and therefore may be used for code review.

Sample data: Download

Reference

- **Evaluator: computer**
  - Program compiled
  - Test 'Zakladni test s parametry podle ukazky': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.000 s (limit: 5.000 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Test meznich hodnot': success
    - result: 100.00 %, required: 50.00 %
    - Total run time: 0.086 s (limit: 5.000 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Test nahodnymi daty': success
    - result: 100.00 %, required: 50.00 %
    - Total run time: 0.022 s (limit: 4.914 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Bonusovy test - manipulatory, mem dbg': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.017 s (limit: 5.000 s)
    - Bonus test - success, evaluation: 120.00 %
  - Test 'Bonusovy test - manipulatory, copyfmt, mem dbg': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.016 s (limit: 4.983 s)
    - Bonus test - success, evaluation: 120.00 %
  - Overall ratio: 144.00 % (= 1.00 * 1.00 * 1.00 * 1.20 * 1.20)
- Total percent: 144.00 %
- Early submission bonus: 0.50
- Total points: 1.44 * (5.00 + 0.50) = 7.92

SW metrics:

	Total	Average	Maximum	Function name
Functions:	**29**	--	--	--
Lines of code:	**298**	**10.28 ± 10.85**	**45**	`CDate::parseDate`
Cyclomatic complexity:	**82**	**2.83 ± 3.84**	**19**	`CDate::parseDate`

2 **2022-03-26 18:08:20** Download

Submission status:	Evaluated
Evaluation:	5.5000

- **Evaluator: computer**
  - Program compiled
  - Test 'Basic test with sample data': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.001 s (limit: 5.000 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Borderline test': success
    - result: 100.00 %, required: 50.00 %
    - Total run time: 0.099 s (limit: 4.999 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Random test': success
    - result: 100.00 %, required: 50.00 %
    - Total run time: 0.051 s (limit: 4.900 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Bonus test - manipulators, mem dbg': Abnormal program termination (Segmentation fault/Bus error/Memory limit exceeded/Stack limit exceeded)
    - Total run time: 0.015 s (limit: 5.000 s)
    - Bonus test - failed, evaluation: No bonus awarded
  - Test 'Bonus test - manipulators, copyfmt, mem dbg': Not tested
    - Bonus test - failed, evaluation: No bonus awarded
  - Overall ratio: 100.00 % (= 1.00 * 1.00 * 1.00)
- Total percent: 100.00 %
- Early submission bonus: 0.50
- Total points: 1.00 * (5.00 + 0.50) = 5.50

SW metrics:

	Total	Average	Maximum	Function name
Functions:	**21**	--	--	--
Lines of code:	**278**	**13.24 ± 19.10**	**95**	`main`
Cyclomatic complexity:	**86**	**4.10 ± 3.49**	**16**	`CDate::operator>>`

1 **2022-03-26 15:03:42**

Submission status: Evaluated
Evaluation: 0.0000

- **Evaluator: computer**
  - ☑ Compile in 'basic' mode failed.

```
In file included from testbed.cpp:417:
tested: In constructor ‘student_2cbda7ea6414e158b38ddf16abb15216::CDate::CDate(student_2cbda7ea6414e158b38ddf16abb15216::USI, student_2cbda7ea6414e15
tested:43:17: error: C++ designated initializers only available with ‘-std=c++2a’ or ‘-std=gnu++2a’ [-Werror=pedantic]
   43 |                 .tm_year = year-1900,
      |                 ^
tested:44:17: error: C++ designated initializers only available with ‘-std=c++2a’ or ‘-std=gnu++2a’ [-Werror=pedantic]
   44 |                 .tm_mon = month - 1, //Decrease (0-11) as ctime month setting
      |                 ^
tested:45:17: error: C++ designated initializers only available with ‘-std=c++2a’ or ‘-std=gnu++2a’ [-Werror=pedantic]
   45 |                 .tm_mday = day
      |                 ^
tested:46:9: error: designator order for field ‘tm::tm_mon’ does not match declaration order in ‘tm’
   46 |         }; m

<skipping 875 B>

s.ios_base::setstate(std::ios::failbit); return iss;}
      |                                                 ^~~~~~~
tested:80:58: error: ‘class std::ios_base’ has no member named ‘setstate’
   80 |                             if (tk3 > 29) {iss.ios_base::setstate(std::ios::failbit); return iss;}
      |                                                          ^~~~~~~
tested:82:58: error: ‘class std::ios_base’ has no member named ‘setstate’
   82 |                             if (tk3 > 28) {iss.ios_base::setstate(std::ios::failbit); return iss;}
      |                                                          ^~~~~~~
tested:86:87: error: ‘class std::ios_base’ has no member named ‘setstate’
   86 |                         case 3: case 5: case 8: case 10: if (tk3 > 30) {iss.ios_base::setstate(std::ios::failbit); return iss;}
      |                                                                                       ^~~~~~~
cc1plus: all warnings being treated as errors
```

- Total percent: 0.00 %
- Early submission bonus: 0.50
- Total points: 0.00 * (5.00 + 0.50) = 0.00