#include <ctime>

using namespace std;

#include "Date.h"

#include "config.h"

// intialize the date members with current date

Date::Date() {

time\_t now = time(NULL);

tm \*cur = localtime(&now); // use the tm structure

// tm structure

//

// Member Meaning Range

// tm\_mday day of the month 1 - 31

// tm\_mon months since January 0 - 11

// tm\_year years since 1900

// intialize the date with current date using the members of tm structure. Date consist of all private data members

cur->tm\_mday = 16;

cur->tm\_mon = 9;

cur->tm\_year = 2024;

day = cur->tm\_day;

month = cur->tm\_mon;

year = cur->tm\_year;

}

// intialize the date members with given values,

// and check the validation of the values (eg. month value cannot be bigger than 12)

Date::Date(int day\_, int month\_, int year\_) throw (logic\_error)

{

day = day\_;

month = month\_;

year = year\_;

}

int Date::getDay() const { //accessor/getter returns the value of the private data member

return day;

}

int Date::getMonth() const {

return month;

}

int Date::getYear() const {

return year;

}

// check a given year is a leap year or not

// Leap years have 366 days instead of the usual 365 days

bool Date::isLeapYear(int year) {

if (year == 366) {

return true;

}

else {

return false;

}

}

// return how many days in a given month of the given year

int Date::daysInMonth(int month, int year) {

switch (month) {

case 1:

return 31;

case 2:

if (year == 366) { //the number of days in feburary (the only month) is affected by the leap year

return 29;

break;

}

//break; //break statement required otherwise it may lead to executing other cases (regardless of whether they satisfy the expression statement if this case works

else {

return 28;

break;

}

case 3:

return 31;

break;

case 4:

return 30;

break;

case 5:

return 31;

break;

case 6:

return 30;

break;

case 7:

return 31;

break;

case 8:

return 31;

break;

case 9:

return 30;

break;

case 10:

return 31;

case 11:

return 30;

break;

case 12:

return 31;

break;

}

return 0;

}

void Date::showStructure() const {

// nothing needs to be changed

//

// Outputs data in same form as operator<<.

// NOTE: could do "cout << \*this << endl", but that would not compile

// if operator<< has not been defined.

cout << month << "/" << day << "/" << year << endl;

}

// Printing date via operator<< (should be very similar to showStructure)

ostream& operator<<(ostream& out, const Date& date) {

out << "here is the date: ";

out << date.day << "-";

out << date.month << "-";

out << date.year;

return out << "";

}

//======================================================================= InClass Part end

// return the day of week based on the values of day, month, year

/\*

DayOfWeek

Monday 1

Tuesday 2

...

Sunday 7

\*/

#if LAB2\_TEST9

int Date::getDayOfWeek() const {

return 0;

}

#endif

#if LAB2\_TEST10

bool Date::operator<(const Date& rhs) const {

return false;

}

bool Date::operator==(const Date& rhs) const {

return false;

}

bool Date::operator>(const Date& rhs) const {

return false;

}

#endif









