**Yibo Zhu**

Reference to run Google Test on Ubuntu:

<https://www.eriksmistad.no/getting-started-with-google-test-on-ubuntu/>

for compile and run the test:

cmake CMakeLists.txt

make

./a.out

A screenshot of a cell phone

Description automatically generated

A close up of a sign

Description automatically generated

A screenshot of text

Description automatically generated

A screenshot of a cell phone screen with text

Description automatically generated

A close up of text on a white background

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone screen with text

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Graduate student question:

* Because private/internal components can’t be accessed directly, we can use the public methods that call them to test it indirectly. For example, to test the private method int compare(const Date&) const; we can use public method Duration duration(const Date&, const Date&); which call compare method inside.

Date::Duration

duration(const Date& date1, const Date& date2) {

int y1 = date1.year;

int y2 = date2.year;

int m1 = date1.month;

int m2 = date2.month;

int d1 = date1.day;

int d2 = date2.day;

// Compute the compare

int order = date1.compare(date2);

if (order == 0)

return Date::Duration(0,0,0);

else if (order > 0) {

// Make date1 precede date2 locally

using std::swap;

swap(y1, y2);

swap(m1, m2);

swap(d1, d2);

}

* Yes, there are some cases that are not covered.

1. The exception structure:

// An exception class

struct DateError : public std::logic\_error {

DateError(const std::string& msg = "")

: std::logic\_error(msg) {}

};

1. Asserts as normal cased are not covered:

int years = y2 - y1;

int months = m2 - m1;

int days = d2 - d1;

assert(years > 0 ||

years == 0 && months > 0 ||

years == 0 && months == 0 && days > 0);

// Do the obvious corrections (must adjust days

// before months!) - This is a loop in case the

// previous month is February, and days < -28.

int lastMonth = m2;

int lastYear = y2;

while (days < 0) {

// Borrow from month

assert(months > 0);

days += Date::daysInPrevMonth(

lastYear, lastMonth--);

--months;

}

if (months < 0) {

// Borrow from year

assert(years > 0);

months += 12;

--years;

}

1. Insertion operator and extraction operator are not covered:

ostream& operator<<(ostream& os, const Date& d) {

char fillc = os.fill('0');

os << setw(2) << d.getMonth() << '-'

<< setw(2) << d.getDay() << '-'

<< setw(4) << setfill(fillc) << d.getYear();

return os;

}

istream& operator>>(istream& is, Date& d) {

is >> d.month;

char dash;

is >> dash;

if (dash != '-')

is.setstate(ios::failbit);

is >> d.day;

is >> dash;

if (dash != '-')

is.setstate(ios::failbit);

is >> d.year;

return is;

}