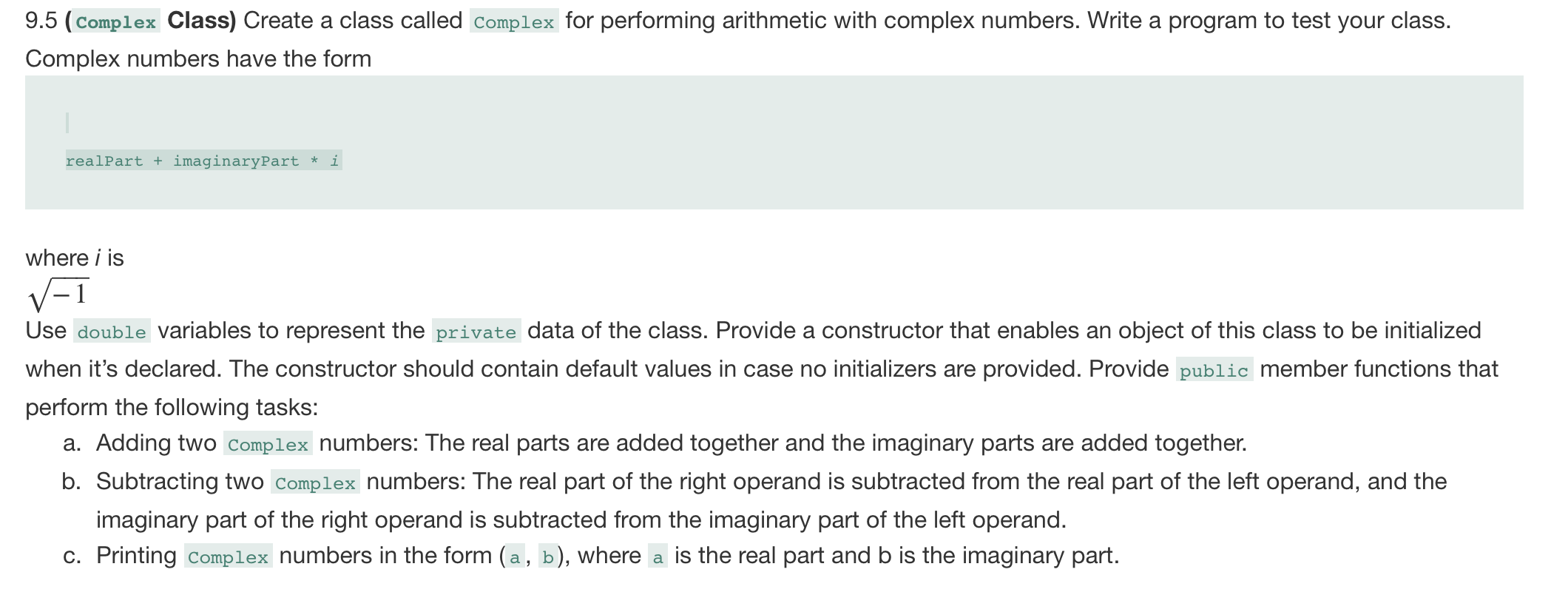
**Problem 9.5**



Code:

**Complex.h**

#ifndef COMPLEX\_H

#define COMPLEX\_H

class Complex

{

private:

double realNum;

double iNum;

public:

explicit Complex(double = 0.0, double = 0.0);

Complex add(Complex);

Complex subtract(Complex);

void print() const;

};

#endif // !COMPLEX\_H

**Complex.cpp**

#include "Complex.h"

#include <iostream>

using namespace std;

Complex::Complex(double r, double i)

{

realNum = r;

iNum = i;

}

Complex Complex::add(Complex obj)

{

Complex temp;

temp.realNum = realNum + obj.realNum;

temp.iNum = iNum + obj.iNum;

return temp;

}

Complex Complex::subtract(Complex obj)

{

Complex temp;

temp.realNum = realNum - obj.realNum;

temp.iNum = iNum - obj.iNum;

return temp;

}

void Complex::print() const

{

cout << "(" << realNum << ", " << iNum << ")";

}

**Test\_complex.cpp**

#include "Complex.h"

#include <iostream>

using namespace std;

int main()

{

Complex a(5.5, -6.3);

Complex b(8.8, 3.3);

Complex c;

cout << "Complex Number Program" << endl;

cout << "a = ";

a.print();

cout<<endl;

cout << "b = ";

b.print();

cout<<endl;

cout << "c = ";

c.print();

cout<<endl;

cout << "Performing an operation : c = a + b" << endl;

c = a.add(b);

cout << "c = ";

c.print();

cout<<endl;

cout << "Performing an operation : c = a - b" << endl;

c = a.subtract(b);

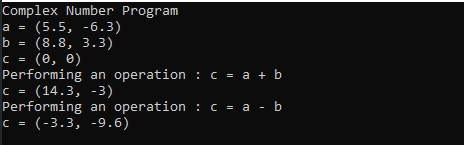
cout << "c = ";

c.print();

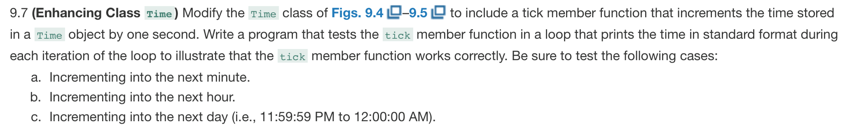
cout<<endl;

return (0);

}



**Problem 9.7**



**Time.h**

#ifndef TIME\_H

#define TIME\_H

// Time class definition

class Time

{

public:

explicit Time(int = 0, int = 0, int = 0); // default constructor

// set functions

void setTime(int, int, int); // set hour, minute, second

void setHour(int); // set hour (after validation)

void setMinute(int); // set minute (after validation)

void setSecond(int); // set second (after validation)

// get functions

unsigned int getHour() const; // return hour

unsigned int getMinute() const; // return minute

unsigned int getSecond() const; // return second

void printUniversal() const; // output time in universal-time format

void printStandard() const; // output time in standard-time format

void tick();

private:

unsigned int hour; // 0 - 23 (24-hour clock format)

unsigned int minute; // 0 - 59

unsigned int second; // 0 - 59

}; // end class Time

#endif

**Time.cpp**

#include <iostream>

#include <iomanip>

#include <stdexcept>

#include "Time.h" // include definition of class Time from Time.h

using namespace std;

// Time constructor initializes each data member

Time::Time(int hour, int minute, int second)

{

setTime(hour, minute, second); // validate and set time

} // end Time constructor

// set new Time value using universal time

void Time::setTime(int h, int m, int s)

{

setHour(h); // set private field hour

setMinute(m); // set private field minute

setSecond(s); // set private field second

} // end function setTime

void Time::tick()

{

second++;

if (second >= 60)

{

minute++;

second = 0;

}

if (minute >= 60)

{

hour++;

minute = 0;

}

if (hour >= 24)

hour = 0;

}

// set hour value

void Time::setHour(int h)

{

if (h >= 0 && h < 24)

hour = h;

else

throw invalid\_argument("hour must be 0-23");

} // end function setHour

// set minute value

void Time::setMinute(int m)

{

if (m >= 0 && m < 60)

minute = m;

else

throw invalid\_argument("minute must be 0-59");

} // end function setMinute

// set second value

void Time::setSecond(int s)

{

if (s >= 0 && s < 60)

second = s;

else

throw invalid\_argument("second must be 0-59");

} // end function setSecond

// return hour value

unsigned int Time::getHour() const

{

return hour;

} // end function getHour

// return minute value

unsigned int Time::getMinute() const

{

return minute;

} // end function getMinute

// return second value

unsigned int Time::getSecond() const

{

return second;

} // end function getSecond

// print Time in universal-time format (HH:MM:SS)

void Time::printUniversal() const

{

cout << setfill('0') << setw(2) << getHour() << ":"

<< setw(2) << getMinute() << ":" << setw(2) << getSecond();

} // end function printUniversal

// print Time in standard-time format (HH:MM:SS AM or PM)

void Time::printStandard() const

{

cout << ((getHour() == 0 || getHour() == 12) ? 12 : getHour() % 12)

<< ":" << setfill('0') << setw(2) << getMinute()

<< ":" << setw(2) << getSecond() << (hour < 12 ? " AM" : " PM");

} // end function printStandard

**Test\_time.cpp**

#include <iostream>

#include <stdexcept>

#include "Time.h" // include definition of class Time from Time.h

using namespace std;

int main()

{

Time t(23, 58, 58); // hour, minute and second specified

for (int i = 0; i < 122; i++)

{

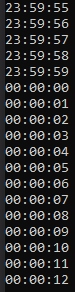
t.tick();

t.printUniversal();

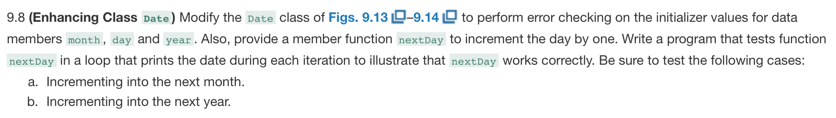
cout << endl;

}

} // end main



**Problem 9.8**



**Date.h**

#pragma once

#ifndef DATE\_H

#define DATE\_H

// class Date definition

class Date

{

public:

explicit Date(int = 1, int = 1, int = 2000); // default constructor

void setDate(int, int, int);

void nextDay();

void print();

private:

unsigned int month;

unsigned int day;

unsigned int year;

}; // end class Date

#endif

**Date.cpp**

#include <iostream>

#include "Date.h" // include definition of class Date from Date.h

using namespace std;

// Date constructor (should do range checking)

Date::Date(int m, int d, int y)

{

setDate(m, d, y);

} // end constructor Date

void Date::setDate(int m, int d, int y)

{

{

if ((m > 0 && m <= 12) && (d > 0 && d <= 31) &&

(y > 0))

{

month = m;

day = d;

year = y;

}

else

throw invalid\_argument("month, day and/or year was out of range");

}

}

bool isLeap(int year)

{

return ((((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0)));

}

void Date::nextDay()

{

day++;

if (day > 30 && (month == 4 || month == 6 || month == 9 || month == 11))

{

month++;

day = 1;

}

else if (month == 2 && ((day > 28 && !isLeap(year)) || (day == 29 && isLeap(year))))

{

month++;

day = 1;

}

else if (day > 31)

{

month++;

day = 1;

}

if (month > 12)

{

year++;

month = 1;

}

}

// print Date in the format mm/dd/yyyy

void Date::print()

{

cout << month << '/' << day << '/' << year;

} // end function print

**Test\_date.cpp**

#include <iostream>

#include "Date.h" // include definition of class Date from Date.h

using namespace std;

int main()

{

Date d(11, 30, 2019);

for (int i = 0; i < 33; i++)

{

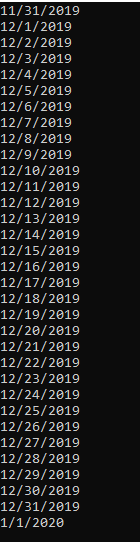
d.nextDay();

d.print();

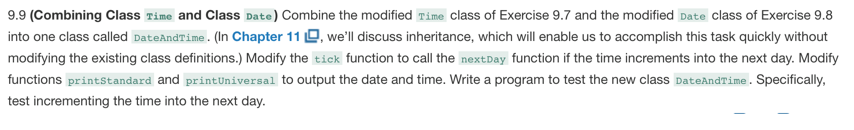
cout << endl;

}

} // end main



**Problem 9.9**



**DateAndTime.h**

#pragma once

#ifndef DATEANDTIME\_H

#define DATEANDTIME\_H

class DateAndTime

{

public:

explicit DateAndTime(int, int, int, int = 1, int = 1, int = 2000); // default constructor

void setDate(int, int, int);

void nextDay();

void setTime(int, int, int); // set hour, minute, second

void setHour(int); // set hour (after validation)

void setMinute(int); // set minute (after validation)

void setSecond(int); // set second (after validation)

unsigned int getHour() const; // return hour

unsigned int getMinute() const; // return minute

unsigned int getSecond() const; // return second

void printStandard() const; // output time in standard-time format

void printUniversal() const;

void tick();

private:

unsigned int month;

unsigned int day;

unsigned int year;

unsigned int hour; // 0 - 23 (24-hour clock format)

unsigned int minute; // 0 - 59

unsigned int second; // 0 - 59

};

#endif // !DATEANDTIME\_H

**DateAndTime.cpp**

#include "DateAndTime.h"

#include <iostream>

#include <stdexcept>

#include <iomanip>

using namespace std;

DateAndTime::DateAndTime(int h, int m, int s, int mnth, int dy, int yr)

{

setDate(mnth, dy, yr);

setTime(h, m, s);

}

void DateAndTime::setDate(int m, int d, int y)

{

if ((m > 0 && m <= 12) && (d > 0 && d <= 31) &&

(y > 0))

{

month = m;

day = d;

year = y;

}

else

throw invalid\_argument("month, day and/or year was out of range");

}

bool isLeap(int year)

{

return ((((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0)));

}

void DateAndTime::nextDay()

{

day++;

if (day > 30 && (month == 4 || month == 6 || month == 9 || month == 11))

{

month++;

day = 1;

}

else if (month == 2 && ((day > 28 && !isLeap(year)) || (day == 29 && isLeap(year))))

{

month++;

day = 1;

}

else if (day > 31)

{

month++;

day = 1;

}

if (month > 12)

{

year++;

month = 1;

}

}

// set new Time value using universal time

void DateAndTime::setTime(int h, int m, int s)

{

setHour(h); // set private field hour

setMinute(m); // set private field minute

setSecond(s); // set private field second

} // end function setTime

// set hour value

void DateAndTime::setHour(int h)

{

if (h >= 0 && h < 24)

hour = h;

else

throw invalid\_argument("hour must be 0-23");

} // end function setHour

// set minute value

void DateAndTime::setMinute(int m)

{

if (m >= 0 && m < 60)

minute = m;

else

throw invalid\_argument("minute must be 0-59");

} // end function setMinute

// set second value

void DateAndTime::setSecond(int s)

{

if (s >= 0 && s < 60)

second = s;

else

throw invalid\_argument("second must be 0-59");

} // end function setSecond

// return hour value

unsigned int DateAndTime::getHour() const

{

return hour;

} // end function getHour

// return minute value

unsigned DateAndTime::getMinute() const

{

return minute;

} // end function getMinute

// return second value

unsigned DateAndTime::getSecond() const

{

return second;

} // end function getSecond

void DateAndTime::printStandard() const

{

cout << month << '/' << day << '/' << year << " " << ((getHour() == 0 || getHour() == 12) ? 12 : getHour() % 12) << ":" << setfill('0') << setw(2) << getMinute()

<< ":" << setw(2) << getSecond() << (hour < 12 ? " AM" : " PM");

} // end function printStandard

void DateAndTime::printUniversal() const

{

cout << month << '/' << day << '/' << year << " " << setfill('0') << setw(2) << getHour() << ":"

<< setw(2) << getMinute() << ":" << setw(2) << getSecond();

}

void DateAndTime::tick()

{

second++;

if (second >= 60)

{

minute++;

second = 0;

}

if (minute >= 60)

{

hour++;

minute = 0;

}

if (hour >= 24)

{

nextDay();

hour = 0;

}

}

**Test\_date.cpp**

#include "DateAndTime.h"

#include <iostream>

#include <iomanip>

using namespace std;

int main()

{

DateAndTime t(23, 59, 53, 12, 31, 2018);

for (int i = 0; i < 10; i++)

{

t.printStandard();

cout << " ";

t.printUniversal();

t.tick();

cout << endl;

}

}

