Ceng 241 Advanced Programming
Midterm
Nov 9, 2004 15.40–17.30
Good Luck!

1 (25 Pts) Create a class **HugeInteger** that uses a 40-element array of digits to store integers as large as 40-digits each.

```
#include <iostream>
using std::cout; using std::endl;
#include "hugeint1.h"
int main()
{
   HugeInteger n1( 7654321 );
   HugeInteger n2( 7891234 );
   HugeInteger n3;
   HugeInteger n4( 5 );
   HugeInteger n5;
   n5 = n1.add( n2 );
   n1.output();
   cout << " + "; n2.output();
   cout << " = "; n5.output();
   cout << "\n\n";
   n5 = n2.subtract( n4 );
   n2.output();
   cout<< " - "; n4.output();
   cout << " = "; n5.output();
   cout << "\n\n";
   if ( n1.isEqualTo( n1 ) == true )
   {n1.output();cout << " is equal ";  n1.output(); cout << "\n\n";}
   if ( n1.isNotEqualTo( n2 ) == true )
   {n1.output();cout << " is not equal to ";n2.output();cout << "\n\n";}
   if ( n2.isGreaterThan( n1 ) == true )
   {n2.output();cout << " is greater than ";n1.output();cout <<"\n\n";}
   if ( n2.isLessThan( n4 ) == true )
   {n4.output();cout << " is less than ";n2.output();cout << "\n\n";}
   if( n4.isLessThanOrEqualTo( n4 ) == true )
   {n4.output();cout << " is less than or equal to ";n4.output();
    cout << "\n\n";}
   if ( n3.isGreaterThanOrEqualTo( n3 ) == true )
   {n3.output();cout << " is greater than or equal to ";n3.output();
    cout << "\n\n";}
   if ( n3.isZero() != true )
   {cout << "n3 contains value ";n3.output();cout << "\n\n";}
return 0;
}
```

- Provide member functions **output, add** and **substract**.
- For comparing **HugeInteger** objects, provide functions **isEqualTo, isNotEqualTo, isGreaterThan, isLessThan, isGreaterThanOrEqualTo** and **isLessThanOrEqualTo** each of these is a *predicate* function that simply returns **true** if the relationship holds between the two huge integers and returns **false** if the relationship does not hold.
- Also, provide a predicate function **isZero**.
- The main function is given above.
- (20 pts) Write a complete program for the class **HugeInteger** with the above capabilities.
- (5 pts) What is the output that the program produces?

```
//class definitions
#ifndef HUGEINT1_H
#define HUGEINT1_H
class HugeInteger {
public:
   HugeInteger( long = 0 );      // conversion/default constructor
   HugeInteger add( const HugeInteger & );//addition operator; HugeInt + HugeInt
   HugeInteger subtract( const HugeInteger & );//subtraction operator; HugeInt - HugeInt
   bool isEqualTo( HugeInteger & );
   bool isNotEqualTo( HugeInteger & );
   bool isGreaterThan(HugeInteger & );
   bool isLessThan( HugeInteger & );
   bool isGreaterThanOrEqualTo( HugeInteger & );
   bool isLessThanOrEqualTo( HugeInteger & );
   bool isZero();
   void output();
   short* getInteger()
   {
   return integer;
   }
private:
   short integer[ 40 ];
};
#endif
//implemetations
#include <iostream>
using std::cout;
#include "hugeint1.h"
// default constructor; conversion constructor that converts
// a long integer into a HugeInteger object
HugeInteger::HugeInteger( long value )
{
   // initialize array to zero
   for ( int i = 0; i < 40; i++ )
```

```
      integer[ i ] = 0;
   // place digits of argument into array
   for ( int j = 39; value != 0 && j >= 0; j-- )
   {
      integer[ j ] = value % 10;
      value /= 10;
   }
}
// addition operator; HugeInteger + HugeInteger
HugeInteger HugeInteger::add( const HugeInteger &op2 )
{
   HugeInteger temp;
   int carry = 0;
   // iterate through HugeInteger
   for ( int i = 39; i >= 0; i-- ) {
   temp.integer[ i ]=integer[ i ] + op2.integer[ i ] + carry;
   // determine whether to carry a 1
     if ( temp.integer[ i ] > 9 )
      { temp.integer[ i ] %= 10;  // reduce to 0-9
        carry = 1;}
     else
        carry = 0;}
return temp;
}
void HugeInteger::output()
{
   int i;
   for ( i = 0; ( integer[ i ] == 0 ) && ( i <= 39 ); i++ )
      ; // skip leading zeros
   if ( i == 40 )
      cout << 0;
   else
      for ( ; i <= 39; i++ )
         cout << integer[ i ];
}
// function to subtract two HugeIntegers
HugeInteger HugeInteger::subtract( const HugeInteger &op2 )
{
   HugeInteger temp;
   int borrow = 0;
   for ( int i = 39; i >= 0; i-- )
   {// determine to add 10 to smaller integer
      if ( integer[i] < op2.integer[i] )
      {temp.integer[ i ]=(integer[i]+10)-op2.integer[i]- borrow;
      borrow = 1;}
      else
      {temp.integer[ i ]=integer[i] - op2.integer[ i ] - borrow;
      borrow = 0;}
   }
return temp;
}
// function that tests if two HugeIntegers are equal
bool HugeInteger::isEqualTo( HugeInteger &x )
{   return integer == x.getInteger();   }
// function that tests if two HugeIntegers are not equal
bool HugeInteger::isNotEqualTo( HugeInteger &x )
{   return !( this->isEqualTo( x ) ); }
// function to test if one HugeInteger is greater than another
bool HugeInteger::isGreaterThan( HugeInteger &x )
{  return integer < x.getInteger();   }
// function that tests if one HugeInteger is less than another
bool HugeInteger::isLessThan( HugeInteger &x )
{  return integer > x.getInteger();   }
// function that tests if one HugeInteger is greater than
// or equal to another
bool HugeInteger::isGreaterThanOrEqualTo( HugeInteger &x )
{  return integer <= x.getInteger();  }
// function that tests if one HugeInteger is less than or
// equal to another
bool HugeInteger::isLessThanOrEqualTo( HugeInteger &x )
{  return integer >= x.getInteger();  }
// function that tests if a HugeInteger is zero
bool HugeInteger::isZero()
{  return ( getInteger() == 0 );      }
//main file
#include <iostream>
using std::cout; using std::endl;
#include "hugeint1.h"
int main()
{
   HugeInteger n1( 7654321 );
   HugeInteger n2( 7891234 );
   HugeInteger n3;
   HugeInteger n4( 5 );
   HugeInteger n5;
   n5 = n1.add( n2 );
   n1.output();
   cout << " + "; n2.output();
   cout << " = "; n5.output();
   cout << "\n\n";
   n5 = n2.subtract( n4 );
   n2.output();
   cout<< " - "; n4.output();
   cout << " = "; n5.output();
   cout << "\n\n";
   if ( n1.isEqualTo( n1 ) == true )
   {n1.output();cout << " is equal ";  n1.output(); cout << "\n\n";}
   if ( n1.isNotEqualTo( n2 ) == true )
   {n1.output();cout << " is not equal to ";n2.output();cout << "\n\n";}
   if ( n2.isGreaterThan( n1 ) == true )
   {n2.output();cout << " is greater than ";n1.output();cout <<"\n\n";}
   if ( n2.isLessThan( n4 ) == true )
   {n4.output();cout << " is less than ";n2.output();cout << "\n\n";}
   if( n4.isLessThanOrEqualTo( n4 ) == true )
   {n4.output();cout << " is less than or equal to ";n4.output();
    cout << "\n\n";}
   if ( n3.isGreaterThanOrEqualTo( n3 ) == true )
   {n3.output();cout << " is greater than or equal to ";n3.output();
    cout << "\n\n";}
   if ( n3.isZero() != true )
   {cout << "n3 contains value ";n3.output();cout << "\n\n";}
return 0;
}
```

```
7654321 + 7891234 = 15545555

7891234 - 5 = 7891229

7654321 is equal 7654321

7654321 is not equal to 7891234

7891234 is greater than 7654321

5 is less than 7891234

5 is less than or equal to 5

0 is greater than or equal to 0

n3 contains value 0
```

2 (25 Pts) Create a class called **Complex** for performing arithmetic with complex numbers. Write a driver program to test your class. Complex numbers have the form

$$realpart + imaginarypart * i$$

where **i** is

$$\sqrt{-1}$$

Use floating–point variables to represent the **private** data of the class. Provide a constructor function that enables an object of this class to be initialized when it is declared. The constructor should contain default values in case no initializers are provided. Provide **public** member functions for each of the following:

- Addition of two **Complex** numbers: The real parts are added together and the imaginary parts are added together
- Substraction of two **Complex** numbers: The real parts are substracted together and the imaginary parts are substracted together
- Multiplication of two **Complex** numbers:

$$(a + ib) * (c + id) = (a * c + i^2 b * d) + i(a * d + b * c)$$

- Printing the results of addition, substraction, and multiplication of **Complex** numbers in the form **(a,b)** where **a** is the real part and **b** is the imaginary part.

```
-----------------------------------------------
#ifndef COMPLEX_H
#define COMPLEX_H
#include <iostream>
using std::cout;
using std::endl;
class Complex{
public:
Complex( double real, double imaginary );
void addition( const Complex & );
void subtraction( const Complex & );
void multiplication( const Complex & );
void printComplex();
void setComplexNumber( double real, double imaginary );
private:
double realPart,imaginaryPart;
};
#endif
-----------------------------------------------
#include "complex.h"
Complex::Complex( double real, double imaginary ){
        setComplexNumber( real, imaginary );}
```

```
void Complex::setComplexNumber( double real, double imaginary ){
            realPart = real;
            imaginaryPart = imaginary;}

void Complex::addition( const Complex &a ){
            realPart+=a.realPart;
            imaginaryPart+=a.imaginaryPart;}

void Complex::subtraction( const Complex &s ) {
            realPart-=s.realPart;
            imaginaryPart-=s.imaginaryPart;}

void Complex::multiplication( const Complex &s ) {
            double temp;
            temp=realPart;
            realPart=realPart*s.realPart - imaginaryPart*s.imaginaryPart;
            imaginaryPart=temp*s.imaginaryPart + imaginaryPart*s.realPart;}

void Complex::printComplex( ){
            cout << '(' << realPart << ", " << imaginaryPart << ')';}
-----------------------------------------------
#include "complex.h"
int main() {
Complex b( 1, 7 ), c( 9, 2 );
b.printComplex(); cout << " + "; c.printComplex();
cout << " = "; b.addition( c ); b.printComplex();
cout << '\n';

b.setComplexNumber( 10, 1 );
c.setComplexNumber( 11, 5 );
b.printComplex(); cout << " - "; c.printComplex();
cout << " = "; b.subtraction( c ); b.printComplex();
cout << endl;

b.setComplexNumber( 19, 15 );
c.setComplexNumber( 13, 8 );
b.printComplex(); cout << " * "; c.printComplex();
cout << " = "; b.multiplication( c ); b.printComplex();
cout << endl;

return 0;
}
```

3 (25 Pts) Implement a **Date** class, where the time will be represented with two forms such as 23/10/2004 or 23/October/2004. The public interface consists of methods to

- Create at least two **Date** objects
- Compute the difference between two dates
- Compute the addition of two dates
- Make necessary controls such as a day shouldn't be negative
- Compare two dates and find the greatest one

```
//Class header file: Date.h
#ifndef Date_h
#define Date_h
class Date{
    private:
        int month,year,day;
        Date validateDate(Date &a);
    public:
        Date(int=1,int=1,int=1900);
        Date setMonth(int);
        Date setYear(int);
        Date setDay(int);
        void setDate(int,int,int);
        int getDay();
        int getYear();
        int getMonth();
        Date addDates(Date& a);
        Date difference_of_dates(Date &a);
        Date findGreatest(Date &a);
        void display();
};
#endif

//class implementation file : Date.cpp
#include "Date.h"
#include <cmath>
#include <iostream>
using std::cout; using std::endl;
Date::Date(int d,int m,int y){
    setDate(d,m,y);}
Date Date::setMonth(int a){
    if(a>=1 && a<=12)
        month=a;
    else
        month=1;
    return *this;}
```

```
Date Date::setYear(int a){
    if(a>=1)
        year=a;
    else
        year=1900;
    return *this;}
Date Date::setDay(int a){int  dayofMonths[]={0,31,28,31,30,31,30,31,31,30,31,30,31};
    if(a<=dayofMonths[month])
        day=a;
    else
        day=1;
    return *this;}
void Date::setDate(int d,int m,int y){
    setMonth(m);
    setDay(d);
    setYear(y);}
int Date::getDay(){   return day;}
int Date::getYear(){   return year;}
int Date::getMonth(){   return month;}
Date Date::addDates(Date& a){
    Date temp;
    temp.day= day+ a.day;
    temp.year=year + a.year;
    temp.month= month + a.month;
    return validateDate(temp);}
Date Date::difference_of_dates(Date &a){
    Date temp;
    temp.day= abs(day - a.day);
    temp.year=abs(year - a.year);
    temp.month= abs(month - a.month);
    return validateDate(temp);}
Date Date::findGreatest(Date &a){
    if(a.year > year)
        return a;
    else if(a.year== year){
        if(a.month > month)
            return a;
        else if(a.month==month){
            if(a.day>=day)
                return a;
            else
                return *this;        }
        else
            return *this;    }
    else
        return *this;}
Date Date::validateDate(Date &a){int dayofMonths[]=
{0,31,28,31,30,31,30,31,31,30,31,30,31};
```

```
    if(a.month>12){
        a.year+=1;
        a.month-=12;    }
    if(a.day>dayofMonths[a.month]){
        a.day-=dayofMonths[a.month];
        a.month+=1;
        if(a.month>12){
            a.year+=1;
            a.month-=12;        }
    }
    return a;}
void Date::display(){
    cout<<day<<"/"<<month<<"/"<<year<<endl;}

//test program
#include "Date.h"
#include<iostream>
using std::cout; using std::endl;
int main(){
    Date a(12,12,2004), b(21,12,2004),c;
    a.display();
    b.display();
    c=a.addDates(b);
    c.display();
    c=a.difference_of_dates(b);
    c.display();
    c=a.findGreatest(b);
    c.display();
    return 0;}
```

4 (25 Pts) Implement a **RationalNumber** class. The public interface consists of methods to

- Create at least two **Rational** number
- The numerator and denominator parts should be integer
- Compute the substraction of two **Rational** numbers by overloading substraction operator
- Compute the addition of two **Rational** numbers by overloading addition operator
- Overload stream insertion >> and stream extraction << operators
- Compare two RationalNumbers by overloading ==,>= and > operators

```
//Class Header file : Rational.h
#ifndef RATIONAL_H
#define RATIONAL_H
#include<iostream>
using std::istream; using std::ostream;
using std::endl;
class Rational{
friend istream& operator>>(istream& input, Rational& a);
friend ostream& operator<<(ostream& output, Rational& a);
    private:
        int num;
        int denom;
    public:
        Rational(int =0,int =1);
        Rational& setNumerator(int);
        Rational& setDenominator(int);
        int getNumerator();
        int getDenominator();
        Rational operator+(Rational &);
        Rational operator-(Rational &);
        bool operator==(Rational &);
        bool operator>=(Rational &);
        bool operator>(Rational &);};
#endif

//class implementation: Rational.cpp
#include "Rational.h"
istream& operator>>(istream& input, Rational& a){
    //input format : 1/2
    input>>a.num;
    input.ignore();
    input>>a.denom;
    return input;}
ostream& operator<<(ostream& output, Rational& a){
    output<<a.num<<"/"<<a.denom;
    return output;}
Rational::Rational(int n,int d){
    setNumerator(n);
    setDenominator(d);}
Rational& Rational::setNumerator(int a){
    num=a;
    return *this;}
Rational& Rational::setDenominator(int a){
    if(a!=0)
        denom=a;
    else
        denom=1;
    return *this;}
int Rational::getNumerator(){
    return num;}
int Rational::getDenominator(){
    return denom;}
Rational Rational::operator+(Rational &a){
    Rational temp;
    if(denom==a.denom){
        temp.num=num+ a.num;
        temp.denom=denom;
    }
    else{
        temp.num=num*a.denom + a.num*denom;
        temp.denom=denom*a.denom;
    }
    return temp;}
Rational Rational::operator-(Rational &a){
    Rational temp;
    if(denom==a.denom){
        temp.num=num- a.num;
        temp.denom=denom;
    }
    else{
        temp.num=num*a.denom - a.num*denom;
        temp.denom=denom*a.denom;
    }
    return temp;}
bool Rational::operator==(Rational &a){
    if((double(a.num)/double(a.denom)) ==(double(num)/double(denom)))
        return true;
    else
        return false;}
bool Rational::operator>=(Rational &a){
    if((double(a.num)/double(a.denom)) >=(double(num)/double(denom)))
        return true;
    else
```

```
        return false;}
bool Rational::operator>(Rational &a){
    if((double(a.num)/double(a.denom)) >(double(num)/double(denom)))
        return true;
    else
        return false;}

//test program
#include <iostream>
using std::cout; using std::cin;
using std::endl;
#include "Rational.h"
int main(){
    Rational a,b,c;
    cout<<"Enter first rational number: ";
    cin>>a;
    cout<<"\nEnter second rational number: ";
    cin>>b;
    cout<<"a="<<a<<"  b="<<b<<endl;
    c=a+b;
    cout<<"a + b="<<c<<endl;
    c=a-b;
    cout<<"a - b="<<c<<endl;
    if(a > b)
        cout<<" a is greater than b"<<endl;
    if(a >= b)
        cout<<" a is greater than equal to b"<<endl;
    if(a==b)
        cout<<" a is equal to b"<<endl;
    return 0;
}
```