Ceng 205 Computer Programming II
Midterm
July 16, 2004 13.00–15.00
Good Luck!

1 (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;
}
```

2 (25 Pts) Create a class **Circle**, which has attribute radius with a default value 1, a **const** attribute PI. It has one constructor and two member functions that calculate perimeter and area of the circle. It has set and get functions for radius. The set function should verify that radius is greater than 0 and less than 50.0.
Hints:

- Perimeter=2*radius*PI
- Area=radius*radius*PI

3 (25 Pts) Create a **SavingsAccount** class. Use a **static** data member to contain the **annualInterestRate** for each of the savers. Each member of the class contains a **private** data member **savingsBalance** indicating the amount the saver currently has on deposit. Provide a **calculateMonthlyInterest** member function that calculates the monthly interest by multiplying the **balance** by **annualInterestRate** divided by 12; this interest should be added to **savingsBalance**. Provide a **static** member function **modifyInterestRate** that sets the **static annualInterestRate** to a new value. Write a driver program to test class **SavingsAccount**. Instantiate two different **savingsAccount** objects, **saver1** and **saver2**, with balances of $2000.00 and $3000.00, respectively. Set **annualInterestRate** to 3%, then calculate the monthly interest and print the new balances for each of the savers. Then set the **annualInterestRate** to 4% and calculate the next month s interest and print the new balances for each of the savers.

```
-----------------------------------------------
#ifndef HEADER_H
#define HEADER_H
class SavingsAccount
{
public:
   SavingsAccount( double b ) { savingsBalance = b >= 0 ? b : 0; }
   void calculateMonthlyInterest();
   static void modifyInterestRate( double );
   void printBalance() const;
private:
   double savingsBalance;
   static double annualInterestRate;
};
#endif
-----------------------------------------------
#include "header.h"
#include <iostream>
using std::cout;
using std::fixed;
#include <iomanip>
using std::setprecision;

// initialize static data member
double SavingsAccount::annualInterestRate = 0.0;

// calculate monthly interest for this savings account
void SavingsAccount::calculateMonthlyInterest()
   { savingsBalance += savingsBalance * ( annualInterestRate / 12.0 ); }

// method for modifying static member variable annualInterestRate
void SavingsAccount::modifyInterestRate( double i )
   { annualInterestRate = ( i >= 0 && i <= 1.0 ) ? i : 0.03; }
```

```
// prints balance of the savings account
void SavingsAccount::printBalance() const
{
   cout << fixed
        << '$' << setprecision( 2 ) << savingsBalance
        << fixed;
}
-----------------------------------------------
#include <iostream>
using std::cout;
using std::endl;
#include <iomanip>
using std::setw;

#include "header.h"
int main()
{
   SavingsAccount saver1( 2000.0 ), saver2( 3000.0 );
   SavingsAccount::modifyInterestRate( .03 );
   cout << "\nOutput monthly balances for one year at .03"
        << "\nBalances: Saver 1 ";
   saver1.printBalance();
   cout << "\tSaver 2 ";
   saver2.printBalance();
   for ( int month = 1; month <= 12; ++month ) {
      saver1.calculateMonthlyInterest();
      saver2.calculateMonthlyInterest();
      cout << "\nMonth" << setw( 3 ) << month << ": Saver 1 ";
      saver1.printBalance();
      cout << "\tSaver 2 ";
      saver2.printBalance();
   }
   SavingsAccount::modifyInterestRate( .04 );
   saver1.calculateMonthlyInterest();
   saver2.calculateMonthlyInterest();
   cout << "\nAfter setting interest rate to .04"
        << "\nBalances: Saver 1 ";
   saver1.printBalance();
   cout << "\tSaver 2 ";
   saver2.printBalance();
   cout << endl;

   return 0;

}
```

4 (25 Pts) Create a class called **Complex** for performing arithmetic with complex numbers. Complex numbers have the form

$$realpart + imaginarypart * i$$

where **i** is

$$\sqrt{-1}$$

- The class must be enable input and output of complex numbers through the overloaded $\gg$ and $\ll$ operators, respectively.
- Overload the multiplication operator to enable multiplication of two complex numbers as in algebra.

```
-------------------------------------------
 #ifndef HEADER_H
#define HEADER_H
#include <iostream>
using std::ostream;
using std::istream;

class Complex {
   friend ostream &operator<<( ostream &, const Complex & );
   friend istream &operator>>( istream &, Complex & );
public:
   Complex( double = 0.0, double = 0.0 );     // constructor
   Complex operator*( const Complex& ) const; // multiplication
private:
   double real;       // real part
   double imaginary;  // imaginary part
};
#endif
-------------------------------------------
#include "header.h"
#include <iostream>
using std::ostream;
using std::istream;

// Constructor
Complex::Complex( double r, double i )
{
   real = r;
   imaginary = i;
} // end Complex constructor


// Overloaded multiplication operator
Complex Complex::operator*( const Complex &operand2 ) const
{
   Complex times;
```

```
   times.real = real * operand2.real + imaginary * operand2.imaginary;
   times.imaginary = real * operand2.imaginary + imaginary *
                  operand2.real;
   return times;
} // end function operator*

ostream& operator<<( ostream &output, const Complex &complex )
{
   output << complex.real << " + " << complex.imaginary << 'i';
   return output;
} // end function operator<<

istream& operator>>( istream &input, Complex &complex )
{
   input >> complex.real;
   input.ignore( 3 );       // skip spaces and +
   input >> complex.imaginary;
   input.ignore( 2 );

   return input;
} // end function operator>>
-------------------------------------------
#include <iostream>

using std::cout;
using std::cin;

#include "header.h"

int main()
{
   Complex x, y( 4.3, 8.2 ), z( 3.3, 1.1 ), k;

   cout << "Enter a complex number in the form: a + bi\n? ";
   cin >> k;
   cout << "x: " << x << "\ny: " << y << "\nz: " << z << "\nk: "
        << k << '\n';
   x = y * z;
   cout << "\nx = y * z:\n" << x << " = " << y << " * " << z << "\n\n";

   return 0;
}
```