**Lab Exercises**

Class:软件1801 Name:李瑞杰 StuID:201816040104

Date:2019/11/9 Score:

**Lab Exercise 1 — Modifying Class Account**

**I Lab Objectives**

In this lab, you will practice:

1. Creating member functions.
2. Invoking functions and receiving return values from functions.
3. Testing a condition using an ifstatement.
4. Outputting variables with stream insertion and the coutobject.

**II Description of the Problem**

Modify class Account to provide a member function called debit that withdraws money from an Account. Ensure that the debit amount does not exceed the Account’s balance. If it does, the balance should be left unchanged and the function should print a message indicating "Debit amount exceeded account balance." Modify class AccountTest to test member function debit.

**III Sample Output**



**IV Your Solution**

**// Lab 1: Account.h**

**// Definition of Account class.**

**class Account**

**{**

**public:**

**Account( int ); // constructor initializes balance**

**void credit( int ); // add an amount to the account balance**

**void debit( int );**

**int getBalance(); // return the account balance**

**private:**

**int balance; // data member that stores the balance**

**}; // end class Account**

**// Lab 1: Account.cpp**

**// Member-function definitions for class Account.**

**#include <iostream>**

**using namespace std;**

**#include "Account.h" // include definition of class Account**

**// Account constructor initializes data member balance**

**Account::Account( int initialBalance )**

**{**

**balance = 0; // assume that the balance begins at 0**

**// if initialBalance is greater than 0, set this value as the**

**// balance of the Account; otherwise, balance remains 0**

**if ( initialBalance> 0 )**

**balance = initialBalance;**

**// if initialBalance is negative, print error message**

**if ( initialBalance< 0 )**

**cout<< "Error: Initial balance cannot be negative.\n" <<endl;**

**} // end Account constructor**

**// credit (add) an amount to the account balance**

**void Account::credit( int amount )**

**{**

**balance = balance + amount; // add amount to balance**

**} // end function credit**

**void Account::debit( int amount)**

**{**

**if( amount <= balance )**

**{**

**balance = balance-amount;**

**}**

**else**

**{**

**cout<<"Debit amount exceeded account balance."<<endl;**

**}**

**}**

**// return the account balance**

**int Account::getBalance()**

**{**

**return balance; // gives the value of balance to the calling function**

**} // end function getBalance**

**// Lab 1: Account.cpp**

**// Member-function definitions for class Account.**

**#include <iostream>**

**using namespace std;**

**#include "Account.h" // include definition of class Account**

**// Account constructor initializes data member balance**

**Account::Account( int initialBalance )**

**{**

**balance = 0; // assume that the balance begins at 0**

**// if initialBalance is greater than 0, set this value as the**

**// balance of the Account; otherwise, balance remains 0**

**if ( initialBalance> 0 )**

**balance = initialBalance;**

**// if initialBalance is negative, print error message**

**if ( initialBalance< 0 )**

**cout<< "Error: Initial balance cannot be negative.\n" <<endl;**

**} // end Account constructor**

**// credit (add) an amount to the account balance**

**void Account::credit( int amount )**

**{**

**balance = balance + amount; // add amount to balance**

**} // end function credit**

**void Account::debit( int amount)**

**{**

**if( amount <= balance )**

**{**

**balance = balance-amount;**

**}**

**else**

**{**

**cout<<"Debit amount exceeded account balance."<<endl;**

**}**

**}**

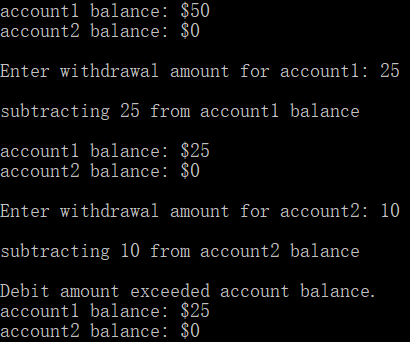
**// return the account balance**

**int Account::getBalance()**

**{**

**return balance; // gives the value of balance to the calling function**

**} // end function getBalance**

****

**Lab Exercise 2 — Modifying class GradeBook**

**I Lab Objectives**

In this lab, you will practice:

1. Declaring a data member.
2. Providing *set* and *get* functions to manipulate a data member’s value.
3. Declaring member functions with parameters.

**II Description of the Problem**

Modify class GradeBook. Include a second string data member that represents thename of the course’s instructor. Provide a *set* function to change the instructor’s name and a *get* function to retrieveit. Modify the constructor to specify *two* parameters—one for thecourse name and one for the instructor’sname. Modify member function displayMessage such that it first outputs the welcome message and coursename, then outputs "This course is presented by: " followed by the instructor’s name. Modify the test applicationto demonstrate the class’s new capabilities.

**III Sample Output**



**IV Your Solution**

**// Lab 2: GradeBook.h**

**// Definition of GradeBook class that stores an instructor's name.**

**#include <string> // program uses C++ standard string class**

**using namespace std;**

**// GradeBook class definition**

**class GradeBook**

**{**

**public:**

**// constructor initializes course name and instructor name**

**GradeBook( string, string );**

**void setCourseName( string ); // function to set the course name**

**string getCourseName(); // function to retrieve the course name**

**void setInstructorName( string );**

**string getInstructorName();**

**void displayMessage(); // display welcome message and instructor name**

**private:**

**string courseName; // course name for this GradeBook**

**string instructorName; // instructor name for this GradeBook**

**}; // end class GradeBook**

**// Lab 2: GradeBook.cpp**

**// Member-function definitions for class GradeBook.**

**#include <iostream>**

**using namespace std;**

**// include definition of class GradeBook from GradeBook.h**

**#include "GradeBook.h"**

**// constructor initializes courseName and instructorName**

**// with strings supplied as arguments**

**GradeBook::GradeBook( string course, string instructor )**

**{**

**setCourseName( course ); // initializes courseName**

**setInstructorName( instructor ); // initialiZesinstructorName**

**} // end GradeBook constructor**

**// function to set the course name**

**void GradeBook::setCourseName( string name )**

**{**

**courseName = name; // store the course name**

**} // end function setCourseName**

**// function to retrieve the course name**

**string GradeBook::getCourseName()**

**{**

**return courseName;**

**} // end function getCourseName**

**void GradeBook::setInstructorName( string name )**

**{**

**instructorName=name;**

**}**

**string GradeBook::getInstructorName()**

**{**

**return instructorName;**

**}**

**// display a welcome message and the instructor's name**

**void GradeBook::displayMessage()**

**{**

**// display a welcome message containing the course name**

**cout<< "Welcome to the grade book for\n" <<getCourseName() << "!"**

**<<endl;**

**cout<<"This course is presented by:" <<getInstructorName() <<endl;**

**} // end function displayMessage**

**// Lab 2: GradeBookTest.cpp**

**// Test program for modified GradeBook class.**

**#include <iostream>**

**using namespace std;**

**// include definition of class GradeBook from GradeBook.h**

**#include "GradeBook.h"**

**// function main begins program execution**

**int main()**

**{**

**// create a GradeBook object; pass a course name and instructor name**

**GradeBookgradeBook("CS101 Introduction to C++ Programming","Sam Smith" );**

**// display welcome message and instructor's name**

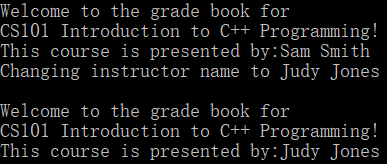
**gradeBook.displayMessage();**

**cout<<"Changing instructor name to Judy Jones"<<"\n"<<endl;**

**gradeBook.setInstructorName("Judy Jones");**

**gradeBook.displayMessage();**

**} // end main**

****

**Lab Exercise 3 — Creating an Employee Class**

**I Lab Objectives**

In this lab, you will practice:

1. Creating a class definition.
2. Declaring data members.
3. Defining a constructor.
4. Defining set and get functions.
5. Writing a test application to demonstrate the capabilities of another class.

**II Description of the Problem**

Create a class called Employee that includes three pieces of information as datamembers—a first name (typestring), a last name (type string) and a monthly salary (type int). Yourclass should have a constructor that initializes the three data members. Provide a *set* and a *get* function for eachdata member. If the monthly salary is not positive, set it to 0. Write a test program that demonstrates class Employee’scapabilities. Create two Employee objects and display each object’s yearly salary. Then give each Employeea 10 percent raise and display each Employee’s yearly salary again.

**III Sample Output**



**IV Your Solution**

// Lab 3: Employee.h  
// Employee class definition.  
  
#include <string> // program uses C++ standard string class  
using namespace std;  
  
// Employee class definition  
class Employee  
{  
public:  
    Employee(string,string,double);  
  
  
    void   setEmployeefirstname( string );  
    string getEmployeefirstname();  
  
    void   setEmployeelastname( string );  
    string getEmployeelastname();  
  
    void   setEmployeemonthlysalary(double);  
    double getEmployeemonthlysalary();  
  
private:  
    string firstname;  
    string lastname;  
    double monthlysalary;  
}; // end class Employee

**// Lab 3: Employee.cpp**

**// Employee class member-function definitions.**

**#include <iostream>**

**using namespace std;**

**#include "Employee.h" // Employee class definition**

**Employee::Employee(string firstname ,string lastname, double monthlysalary)**

**{**

**setEmployeefirstname( firstname );**

**setEmployeelastname( lastname );**

**setEmployeemonthlysalary(monthlysalary);**

**}**

**void Employee::setEmployeefirstname(string Firstname)**

**{**

**firstname=Firstname;**

**}/\* Define a set function for the first name data member. \*/**

**string Employee::getEmployeefirstname()**

**{**

**return firstname;**

**}**

**void Employee::setEmployeelastname(string Lastname)**

**{**

**lastname=Lastname;**

**}**

**string Employee::getEmployeelastname()**

**{**

**return lastname;**

**}**

**void Employee::setEmployeemonthlysalary(double a)**

**{**

**monthlysalary=a;**

**}**

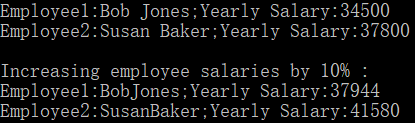
**double Employee::getEmployeemonthlysalary()**

**{**

**return monthlysalary;**

**}**

**// Lab 3: EmployeeTest.cpp  
// Create and manipulate two Employee objects.  
#include <iostream>  
using namespace std;  
  
#include "Employee.h" // include definition of class Employee  
  
// function main begins program execution  
int main()  
{  
    string Firstname1;  
    string Lastname1;  
    double Monthlysalary1;  
  
    string Firstname2;  
    string Lastname2;  
    double Monthlysalary2;  
    double a,b;  
  
   cout<<"Firstname1 is:"<<endl;  
   cin>>Firstname1;  
   cout<<"Lastname1 is:"<<endl;  
   cin>>Lastname1;  
   cout<<"Monthlysalary1 is:"<<endl;  
   cin>>Monthlysalary1;  
  
   cin>>Firstname2;  
   cout<<"Lastname2 is:"<<endl;  
   cin>>Lastname2;  
   cout<<"Monthlysalary2 is:"<<endl;  
   cin>>Monthlysalary2;  
  
  
        Employee employee1(Firstname1,Lastname1,Monthlysalary1);  
  
        employee1.setEmployeefirstname(Firstname1);  
        employee1.setEmployeelastname(Lastname1);  
        employee1.setEmployeemonthlysalary(Monthlysalary1);  
        cout<<"Firstname2 is:"<<endl;  
  
        Employee employee2(Firstname2,Lastname2,Monthlysalary2);  
  
        employee2.setEmployeefirstname(Firstname2);  
        employee2.setEmployeelastname(Lastname2);  
        employee2.setEmployeemonthlysalary(Monthlysalary2);  
  
   a=employee1.getEmployeemonthlysalary();  
   b=employee2.getEmployeemonthlysalary();  
  
   a=1.1\*a;  
   b=1.1\*b;  
   cout<<"Increasing employee slaries by 10%"<<endl;  
   cout<<employee1.getEmployeefirstname() << employee1.getEmployeelastname()<<"Yearly salary is:"<<a<<endl;  
   cout<<employee2.getEmployeefirstname()<<employee2.getEmployeelastname()<<"Yearly salary is:"<<b<<endl;  
} // end main**

****

**Lab Exercise 4 — Complex Numbers**

**I Lab Objectives**

In this lab, you will practice:

1. Creating new data types by writing class definitions.
2. Defining member functions of programmer-defined classes.
3. Instantiating objects from programmer-defined classes.
4. Calling member functions of programmer-defined classes.

The follow-up questions and activities will also give you practice:

1. Initializing programmer-defined class data members with class constructors.

**II Description of the Problem**

Create a class called Complex for performing arithmetic with complex numbers. Write a program to test your class.

Complex numbers have the form：

realPart + imaginaryPart \* *i*

where*i* is

Use double variables to represent the private data of the class. Provide a constructor 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 memberfunctions that perform the following tasks:

1) Adding two Complex numbers: The real parts are added together and the imaginary parts are added together.

2) Subtracting two Complex numbers: The real part of the right operand is subtracted from the real part of the left operand and the imaginary part of the right operand is subtracted from the imaginary part of the left operand.

3) Printing Complex numbers in the form (a, b) where a is the real part and b is the imaginary part.

**III Sample Output**



**IV Your Solution**

**// Lab 4: Complex.h  
#ifndef COMPLEX\_H  
#define COMPLEX\_H  
  
/\* Write class definition for Complex \*/  
class Complex  
{  
    public:  
    Complex(double,double);  
    void setComplexNumber( double , double  );  
    Complex add(const Complex &);  
    Complex subtract(const Complex &);  
    void printComplex();  
    private:  
    double    realPart;  
    double    imaginaryPart;  
};  
#endif**

**// Lab 4: Complex.cpp  
// Member-function definitions for class Complex.  
#include <iostream>  
using namespace std;  
  
#include "Complex.h"  
  
Complex::Complex( double real, double imaginary )  
{  
   setComplexNumber( real, imaginary );  
} // end Complex constructor  
  
Complex Complex::add( const Complex &right )  
{  
  realPart+=right.realPart;  
  imaginaryPart+=right.imaginaryPart;  
  return Complex(realPart,imaginaryPart); /\* Write a statement to return a Complex object. Add  
      the realPart of right to the realPart of this Complex  
      object and add the imaginaryPart of right to the  
      imaginaryPart of this Complex object \*/  
} // end function add  
  
Complex Complex::subtract( const Complex &right )  
{  
realPart-=right.realPart;  
  imaginaryPart-=right.imaginaryPart;  
return Complex(realPart,imaginaryPart);  /\* Write a statement to return a Complex object. Subtract  
      the realPart of right from the realPart of this Complex  
      object and subtract the imaginaryPart of right from  
      the imaginaryPart of this Complex object \*/  
} // end function subtract  
  
void Complex::printComplex()  
{  
   cout << '(' << realPart << ", " << imaginaryPart << ')';  
} // end function printComplex  
  
void Complex::setComplexNumber( double rp, double ip )  
{  
   realPart = rp;  
   imaginaryPart = ip;  
} // end function setComplexNumber**

**// Lab 4: ComplexTest.cpp  
#include <iostream>  
using namespace std;  
  
#include "Complex.cpp"  
  
int main()  
{  
   Complex a( 1, 7 ), b( 9, 2 ), c(1,1); // create three Complex objects  
  
   a.printComplex(); // output object a  
   cout << " + ";  
   b.printComplex(); // output object b  
   cout << " = ";  
   c = a.add( b ); // invoke add function and assign to object c  
   c.printComplex(); // output object c  
  
   cout << '\n';  
   a.setComplexNumber( 10, 1 ); // reset realPart and  
   b.setComplexNumber( 11, 5 ); // and imaginaryPart  
   a.printComplex(); // output object a  
   cout << " - ";  
   b.printComplex(); // output object b  
   cout << " = ";  
   c = a.subtract( b ); // invoke add function and assign to object c  
   c.printComplex(); // output object c  
cout << endl;  
} // end main**

****

**VFollow-Up Questions and Activities**

1. Why do you think const was used in the parameter list of add and subtract?

提高效率，节约空间。

1. Can add and subtract’s parameters be passed by value instead of by reference? How might this affect thedesign of class Complex? Write a new class definition that illustrates how the parameters would be passed byvalue.

**class Complex**

**{**

**public:**

**Complex( double real, double imaginary )**

**{**

**setComplexNumber( real, imaginary );**

**}**

**Complex()**

**{**

**realPart=0;**

**imaginaryPart=0;**

**}**

**void add( double r1,double i1,double r2,double i2)**

**{**

**realPart=r1+r2;**

**imaginaryPart=i1+i2;**

**}**

**void subtract(double r1,double i1,double r2,double i2)**

**{**

**realPart=r1-r2;**

**imaginaryPart=i1-i2;**

**}**

**void printComplex()**

**{**

**cout<< '(' <<realPart<< ", " <<imaginaryPart<< ')';**

**}**

**void setComplexNumber( double rp, double ip )**

**{**

**realPart = rp;**

**imaginaryPart = ip;**

**}**

**double getRealpart()**

**{**

**return realPart;**

**}**

**double getImaginaryPart()**

**{**

**return imaginaryPart;**

**}**

**private:**

**double realPart;**

**double imaginaryPart;**

**};**

1. Declare a Complex number, as follows, without passing any arguments to the constructor. What happens?Does the default constructor get called?

Complex a;

没有真正在内存中创建对象。**Lab Exercise 5 — Dates**

**I Lab Objectives**

In this lab, you will practice:

1. Using access functions and utility functions so that it is not necessary for non-member functions to be able to access a class’ data members.

The follow-up questions and activities also will give you practice:

1. Overloading constructors and using default arguments with constructors.
2. Defining a destructor.

**II Description of the Problem**

Modify the Dateclass to provide a member function nextDayto increment the day by one. The Dateobject should always remain in a consistent state. Write a program that tests function nextDayin a loop that prints the date during each iteration to illustrate that the nextDayfunction works correctly. Be sure to test the following cases:

1. Incrementing into the next month.
2. Incrementing into the next year.

**III Sample Output**



**IV Your Solution**

**// Lab 5: DateTest.cpp**

**#include <iostream>**

**using namespace std;**

**#include "Date.h" // include definitions of class Date**

**int main()**

**{**

**const int MAXDAYS = 16;**

**Date d( 12, 24, 2004 ); // instantiate object d of class Date**

**// output Date object d's value**

**for ( int loop = 1; loop <= MAXDAYS; ++loop )**

**{**

**d.print(); // invokes function print**

**d.nextDay();**

**} // end for**

**cout << endl;**

**} // end main**

**// Lab 5: Date.cpp**

**// Member-function definitions for class Date.**

**#include <iostream>**

**using namespace std;**

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

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

**{**

**setDate( m, d, y ); // sets date**

**} // end Date constructor**

**void Date::setDate( int mo, int dy, int yr )**

**{**

**setMonth( mo ); // invokes function setMonth**

**setDay( dy ); // invokes function setDay**

**setYear( yr ); // invokes function setYear**

**} // end function setDate**

**void Date::setDay( int d )**

**{**

**if ( month == 2 && leapYear() )**

**day = ( d <= 29 && d >= 1 ) ? d : 1;**

**else**

**day = ( d <= monthDays() && d >= 1 ) ? d : 1;**

**} // end function setDay**

**void Date::setMonth( int m )**

**{**

**month = m <= 12 && m >= 1 ? m : 1; // sets month**

**} // end function setMonth**

**void Date::setYear( int y )**

**{**

**year = y >= 1900 ? y : 1900; // sets year**

**} // end function setYear**

**int Date::getDay()**

**{**

**return day;**

**} // end function getDay**

**int Date::getMonth()**

**{**

**return month;**

**} // end function getMonth**

**int Date::getYear()**

**{**

**return year;**

**} // end function getYear**

**void Date::print()**

**{**

**cout << month << '-' << day << '-' << year << '\n'; // outputs date**

**} // end function print**

**void Date::nextDay()**

**{**

**int day1,month1,year1;**

**day1=getDay()+1;**

**month1=getMonth();**

**year1=getYear();**

**while(day1>monthDays()&&month1++<=12)**

**{**

**day1=1;**

**month1++;**

**cout << month1 << '-' << day1<< '-' << year1 << '\n';**

**}**

**while(day1<monthDays())**

**{**

**cout << month1 << '-' << day1<< '-' << year1 << '\n';**

**}**

**while(day1>monthDays()&&month1++>12)**

**{**

**day1=1;**

**month1=1;**

**year1++;**

**cout << month1 << '-' << day1<< '-' << year1 << '\n';**

**}**

**}**

**bool Date::leapYear()**

**{**

**if ( getYear() % 400 == 0 || ( getYear() % 4 == 0 && getYear() % 100 != 0 ) )**

**return true; // is a leap year**

**else**

**return false; // is not a leap year**

**} // end function leapYear**

**bool Date::leapYear()**

**{**

**if ( getYear() % 400 == 0 || ( getYear() % 4 == 0 && getYear() % 100 != 0 ) )**

**return true; // is a leap year**

**else**

**return false; // is not a leap year**

**} // end function leapYear**

**int Date::monthDays()**

**{**

**const int days[ 12 ] =**

**{ 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };**

**return getMonth() == 2 && leapYear() ? 29 : days[ getMonth() - 1 ];**

**} // end function monthDays**

**// Lab 5: Date.h**

**#ifndef DATE\_H**

**#define DATE\_H**

**class Date**

**{**

**public:**

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

**void print(); // print function**

**void setDate( int, int, int ); // set month, day, year**

**void setMonth( int ); // set month**

**void setDay( int ); // set day**

**void setYear( int ); // set year**

**int getMonth(); // get month**

**int getDay(); // get day**

**int getYear(); // get year**

**void nextDay();**

**private:**

**int month; // 1-12**

**int day; // 1-31 (except February(leap year), April, June, Sept, Nov)**

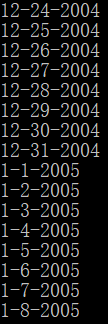
**int year; // 1900+**

**bool leapYear(); // leap year**

**int monthDays(); // days in month**

**}; // end class Date**

**#endif**

****

**V Follow-Up Questions and Activities**

1. The Date class has only one constructor. Is it possible to have more than one constructor?
2. What happens when a member function that takes no arguments is called without the parentheses (i.e.,dateObject.nextDay)?
3. Write a destructor for the Date class. The destructor should print text indicating that the destructor for the Date class was called successfully.
4. In main, try to change d’s year to 2003 using an assignment statement. Do not call function setYear. Whathappens? Are you able to change the value?

**1可以**

**2编译报错**

**3 Date::Date()**

**{**

**Cout<<”The destructor for the Date class was called successfully.”<<endl;**

**}**

**4不能**

**Lab Exercise 6 — Simple Calculator**

**I Lab Objectives**

In this lab, you will practice:

1. Using classes to create a data type Simple Calculator capable of performing arithmetic operations.
2. Creating const member functions to enforce the principle of least privilege.

The follow-up questions and activities also will give you practice:

1. Using constructors to specify initial values for data members of a programmer-defined class.

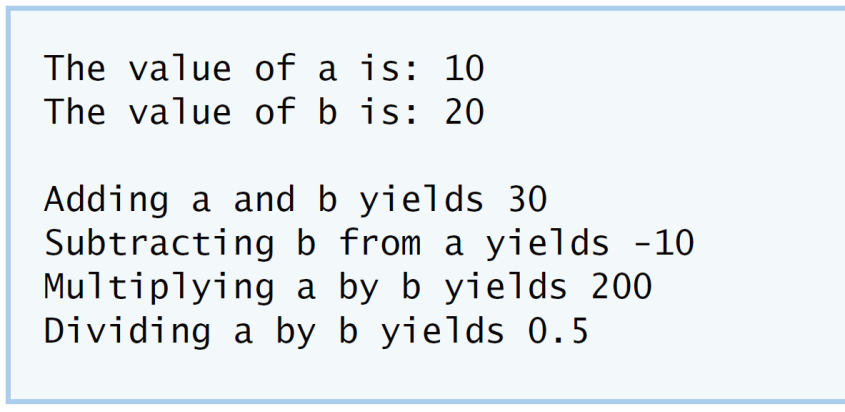
**II Description of the Problem**

Write a SimpleCalculator class that has public methods for adding, subtracting, multiplying and dividing two doubles. A sample call is as follows:

double answer = sc.add( a, b );

Object sc is of type SimpleCalculator. Member function add returns the result of adding its two arguments.

**III Sample Output**



**IV Your Solution**

**// Lab Exercise 6: SimpleCalculator.h**

**// class SimpleCalculator definition**

**class SimpleCalculator**

**{**

**public:**

**SimpleCalculator(double=0,double=0);**

**void setSimpleCalculator(double,double);**

**void setBefore(double);**

**void setLast(double);**

**double getBefore()const;**

**double getLast()const;**

**/\* Write prototype for add member function \*/**

**double add( double, double ) const;**

**double subtract( double, double ) const;**

**double multiply( double, double ) const;**

**double divide( double, double ) const;**

**/\* Write prototype for divide member function \*/**

**private:**

**double right;**

**double left;**

**}; // end class SimpleCalculator**

**// Lab Exercise 6: SimpleCalculator.cpp**

**#include <iostream>**

**using namespace std;**

**#include "SimpleCalculator.h"**

**SimpleCalculator::SimpleCalculator(double right,double left)**

**{**

**setSimpleCalculator(right,left);**

**}**

**void SimpleCalculator::setSimpleCalculator(double right,double left)**

**{**

**setBefore(right);**

**setLast(left);**

**}**

**void SimpleCalculator::setBefore(double a)**

**{**

**right=a;**

**}**

**void SimpleCalculator::setLast(double b)**

**{**

**left=b;**

**}**

**double SimpleCalculator::getBefore()const**

**{**

**return right;**

**}**

**double SimpleCalculator::getLast()const**

**{**

**return left;**

**}**

**// function subtract definition**

**double SimpleCalculator::add( double a, double b ) const**

**{**

**return a + b;**

**} // end function multiply**

**double SimpleCalculator::subtract( double a, double b ) const**

**{**

**return a - b;**

**} // end function subtract**

**// function multiply definition**

**double SimpleCalculator::multiply( double a, double b ) const**

**{**

**return a \* b;**

**} // end function multiply**

**double SimpleCalculator::divide( double a, double b) const**

**{**

**return a/b;**

**}**

**/\* Write definition for divide member function \*/**

**#include <iostream>**

**using namespace std;**

**#include "SimpleCalculator.h"**

**using namespace std;**

**int main()**

**{**

**SimpleCalculator sc(10,20);**

**cout<<"the value of a is :"<<sc.getBefore()<<endl;**

**cout<<"the value of b is :"<<sc.getLast()<<endl;**

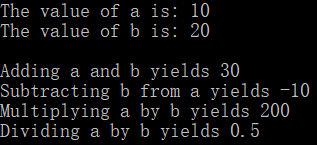
**cout<<"adding a and b yields :"<<sc.add(10,20)<<endl;**

**cout<<"Subtracting b from a yields :"<<sc.subtract(10,20)<<endl;**

**cout<<"Multiplying a by b yields :"<<sc.multiply(10,20)<<endl;**

**cout<<"Dividing b from a yields :"<<sc.divide(10,20)<<endl;**

**}**

****

**V Follow-Up Questions and Activities**

1. Why doesn’t the SimpleCalculator class have a constructor?
2. Why are no private data members needed for class SimpleCalculator?
3. Modify your class so that SimpleCalculatorhas a private data member called answer. After performing an operation, assign the result to answer. Add a member function named getAnswer to retrieve the result of the last arithmetic operation performed by the object. Also, add a constructor for class SimpleCalculator that initializes the value of answer to 0.
4. Modify the program so that the SimpleCalculator class has an input member function that allows the userto input two doubles. The function should then store the values that were input in private data members.Use these two values for each of the arithmetic calculations. Create two constructors for this class, one thattakes no arguments and initializes a andb to 0 and another that takes two doubles and initializes a and b tothose values. Finally, create a member function printValues that displays the values of a andb. A segmentof the driver program might now look like this:



**1不需要初始化，系统将分配默认参数。**

**2不需要维护任何私有数据来执行其算术计算**

**3 class SimpleCaculator**

**{**

**public:**

**SimpleCaculator()**

**{**

**answer = 0;**

**}**

**double add(double a, double b)**

**{**

**answer = a + b;**

**return a + b;**

**}**

**double subtract(double a, double b)**

**{**

**answer = a - b;**

**return a - b;**

**}**

**double multiply(double a, double b)**

**{**

**answer = a \* b;**

**return a \* b;**

**}**

**double divide(double a, double b)**

**{**

**answer = a / b;**

**return a / b;**

**}**

**double getAnswer()**

**{**

**return answer;**

**}**

**private:**

**double answer;**

**};**

**4 class SimpleCaculator**

**{**

**public:**

**SimpleCaculator()**

**{**

**answer = 0;**

**a=0;**

**b=0;**

**}**

**void input()**

**{**

**cin>>a>>b;**

**}**

**void privntValues()**

**{**

**cout<<"a is "<<a<<" and b is "<<b<<endl;**

**}**

**double add(double a, double b)**

**{**

**answer = a + b;**

**return a + b;**

**}**

**double subtract(double a, double b)**

**{**

**answer = a - b;**

**return a - b;**

**}**

**double multiply(double a, double b)**

**{**

**answer = a \* b;**

**return a \* b;**

**}**

**double divide(double a, double b)**

**{**

**answer = a / b;**

**return a / b;**

**}**

**double getAnswer()**

**{**

**return answer;**

**}**

**private:**

**double answer;**

**double a,b;**

**};**

**\*Lab Exercise 7 — Integer Set**

**I Lab Objectives**

In this lab, you will practice:

1. Using classes to create a data type, IntegerSet, capable of storing a set of integers
2. Using dynamic memory allocation with the new and delete operators

The follow-up questions and activities also will give you practice:

1. Using destructors to deallocate memory that was dynamically allocated.

**II Description of the Problem**

Create class IntegerSet for which each object can hold integers in the range 0 through 100. A set is represented internally as an array of ones and zeros. Array element a[ i ] is 1 if integer *i* is in the set. Array element a[ j ] is 0 if integer *j* is not in the set. The default constructor initializes a set to the so-called “empty-set,” i.e., a set whose array representation contains all zeros.

Provide member functions for the common set operations. For example, aunionOfSets member function (already provided) creates a third set that is the set-theoretic union of two existing sets (i.e., an element of the third array’s is set to 1 if that element is 1 in either or both of the existing sets, and an element of the third set’s array is set to 0 if that element is 0 in each of the existing sets).

Provide an intersectionOfSetsmember function which creates a third set which is the set-theoretic intersection of two existing sets (i.e., an element of the third set’s array is set to 0 if that element is 0 in either or both of the existing sets, and an element of the third set’s array is set to 1 if that element is 1 in each of the existing sets).

An insertElement member function (already provided) inserts a new integer k into a set (by setting a[ k ] to 1 ). Provide a deleteElement member function that deletes integer m (by setting a[ m ] to 0 ).

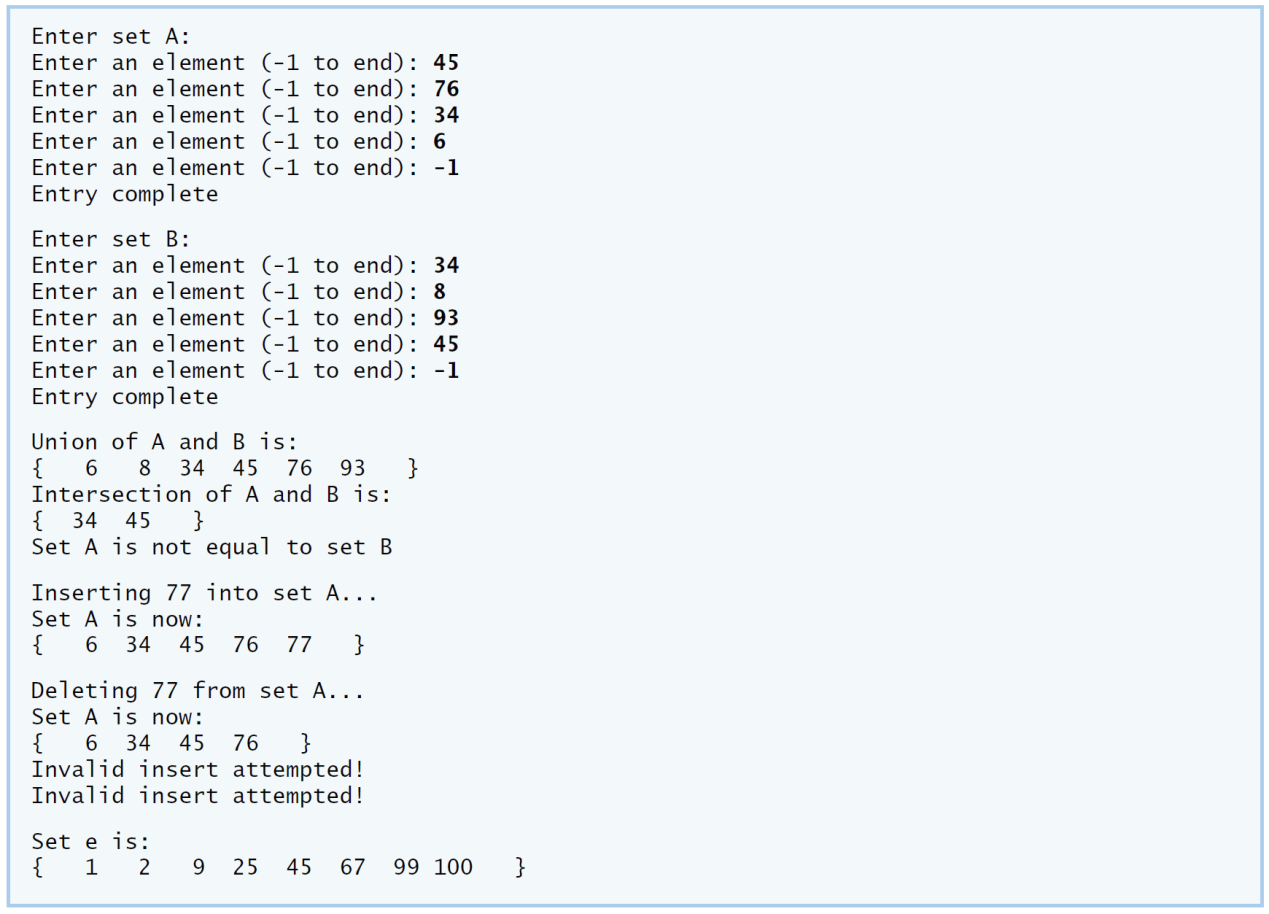
A printSet member function (already provided) prints a set as a list of numbers separated by spaces. Print only those elements which are present in the set (i.e., their position in the array has a value of 1 ). Print --- for an empty set.

Provide anisEqualTo member function that determines whether two sets are equal.

Provide an additional constructor that receives an array of integers and the size of that array and uses the array to initialize a set object.

Now write a driver program to test your IntegerSet class. Instantiate several IntegerSet objects. Test that all your member functions work properly.

**III Sample Output**



**IV Your Solution**

**V Follow-Up Questions and Activities**

1. Why might it be advantageous for the set array to be allocated dynamically using new [], if the IntegerSetclass were to be used for more general sets?