ECE 462 Exam 2

6:30-7:30PM, October 20, 2010

I will not receive nor provide aid to any other student for this exam.

Signature:

You must sign here. Otherwise, the exam is not graded.

This exam is printed **double sides**.

Write your answers next to the questions. If you need more space, you can use the two blank pages.

This is an *open-book, open-note* exam. You can use any book or note or program printouts.

Please turn off your cellular phone **now** .

Two outcomes are tested in this exam. To pass each outcome, you must receive 50% of the points.

- Outcome 5: ability to overload operators in C++.
- Outcome 6: understanding of the difference between function overloading and function overriding.

You need to obtain 50% of the points in each outcome to pass the outcome. If a program has a syntax error, **point out** which line causes the error.

If a question has a numeric answer, you can write the *procedure* without the final result. For example, you can write "1 + 2" instead of "3".

Contents

1 C++ Operator / (Outcome 5, 2 points) 4

2 C++ Operator ++ (Outcome 5, 2 points) 6

3 C++ Operator [] (Outcome 5, 2 points) 8

4 C++ Overloading and Overriding (Outcome 6, 3 points) 10

5 Java Overloading and Overriding (Outcome 6, 3 points) 13

6 Group Programming Assignment (3 point) 15

Passed Outcomes: 5 6

Total Score:

This page is blank.

1 C++ Operator / (Outcome 5, 2 points)

Overload the division operator for complex. Division is defined as

$$\frac{a+bi}{c+di} = \frac{(a+bi)(c-di)}{(c+di)(c-di)} = \frac{(ac+bd)+(bc-ad)i}{c^2+d^2} \tag{1}$$

```
#include <iostream>
using namespace std;

class MyComplex
{
public:
  MyComplex(double r, double i) : re(r), im(i)
  {}
  //
  // <--- FIX ME
  //
  // declare operator / as a member




  friend ostream& operator<< (ostream&, const MyComplex&);
private:
  double re, im;
};

//
// <--- FIX ME
//
// implement operator /
```

```
ostream & operator<< (ostream& os, const MyComplex& c)
{
  os << "(" << c.re << ", " << c.im << ")" << endl;
  return os;
}

int main()
{
  MyComplex first(3, 4);
  MyComplex second(2, 9);
  MyComplex third = first / second;
  cout << first;
  cout << second;
  cout << third;
  return 0;
}
```

Answer:

```
class MyComplex
{
public:
  MyComplex(double r, double i) : re(r), im(i)
  {}
  MyComplex operator / (const MyComplex & denominator) const;
  friend ostream& operator<< (ostream&, const MyComplex&);
private:
  double re, im;
};

MyComplex MyComplex::operator / (const MyComplex & denominator) const
{
  double r = denominator.re;
  double i = denominator.im;
  double de = r * r + i * i;
  double d1 = (re * r + im * i) / de;
  double d2 = (im * r - re * i) / de;
  return MyComplex(d1, d2);
}
```

2 C++ Operator ++ (Outcome 5, 2 points)

Implement the prefix increment operator ++.

```
#include <iostream>
#include <cstdlib>
using namespace std;
class MyInt
{
private:
  int value;
public:
  MyInt(int v = 0) { value = v; }
  //
  // <--- FIX ME
  //
  // declare prefix operator ++




  friend ostream& operator<<(ostream& os, const MyInt& s);
};
//
// <--- FIX ME
//
// implement prefix operator ++




ostream& operator<<(ostream& os, const MyInt& s)
{
  os << s.value;
  return os;
}

int main()
{
  MyInt s1 (3);
  MyInt s2 = ++ s1;
```

```
  cout << s1 << endl;
  cout << s2 << endl;
  return 0;
}
```

Answer:

```
class MyInt
{
  int value;
public:
  MyInt(int v = 0) { value = v; }
  MyInt & operator++();
  friend ostream& operator<<(ostream& os, const MyInt& s);
};

//prefix increment operator:
inline MyInt& MyInt::operator++()
{
  value ++;
  return *this;
}
```

3 C++ Operator [] (Outcome 5, 2 points)

Overload the operator [] for MyVector.

```
#include <iostream>
using namespace std;

class MyVector
{
private:
  int * data;
  int length;
  void construct(const int* d, int s)
  {
    length = s;
    data = new int[s];
    for (int i = 0; i < s; i ++)
      { data[i] = d[i]; }
  }
public:
  MyVector(const int* d, int s)
  {
    construct(d, s);
  }

  ~MyVector()
  { delete[] data; }

  // copy constructor
  MyVector(const MyVector & v)
  {
    construct(v.data, v.length);
  }
  // assignment operator
  MyVector& operator = (const MyVector& v)
  {
    if (this == & v)
      { return *this; }
    delete[] data;
    length = v.length;
    data = new int[length];
    for (int i = 0; i < v.length; i ++)
      { data[i] = v.data[i]; }
    return *this;
  }
  //
```

```
  // <--- FIX ME
  //
  // operator [] to return an element
  // checking the index not necessary









  friend ostream & operator << (ostream &, const MyVector &);
};

ostream & operator<< (ostream & os, const MyVector & v)
{
  for (int i = 0; i < v.length; i ++)
    { os << v.data[i] << " "; }
  return os;
}

int main(int argc, char * argv[])
{
  const int size = 5;
  int d[size] = { 0, 1, 2, 3, 4};
  MyVector s1 (d, size);
  MyVector s2 = s1;
  s1[2] = 10;            // use operator []
  cout << s1 << endl;    // 0 1 10 3 4
  int a = s2[3];         // use operator []
  cout << a << endl;     // 3
  return 0;
}
```

Answer:

```
  int & operator [] (int i)
  {
    cout << "int & operator [] (int i)" << endl;
    return data[i];
  }
```

4 C++ Overloading and Overriding (Outcome 6, 3 points)

What is the output of this program?

```
#include <iostream>
#include <string>
using namespace std;
class B1
{
public:
  void f1(int a) // CAUTION: not virtual
  { cout << "B1:f1(int)" << endl; }
  virtual void f1(const char *  b)
  { cout << "B1:f1(const char * )" << endl; }
};

class D1 : public B1
{
public:
  virtual void f1(int a)
  { cout << "D1:f1(int)" << endl; }
};

class D2 : public D1
{
public:
  virtual void f1(const char *  b)
  { cout << "D2:f1(const char * )" << endl; }
};

class B2
{
public:
  virtual void f2(B1 * b)
  { cout << "B2:f2(B1 *) "; b -> f1(462); }
  virtual void f2(D2 * d)
  { cout << "B2:f2(D2 *) "; d -> f1("ece"); }
};

class D21 : public B2
{
public:
  virtual void f2(D1 * d)
  { cout << "D21:f2(D1 *) "; d -> f1(462); }
  virtual void f2(D2 * d)
  { cout << "D21:f2(D2 *) "; d -> f1("ece"); }
```

```
};

class D22 : public D21
{
public:
  virtual void f2(B1 * d)
  { cout << "D22:f2(B1 *) "; d -> f1(462); }
  virtual void f2(D2 * d)
  { cout << "D22:f2(D2 *) "; d -> f1("ece"); }
};

int main(int argc, const char * argv[])
{
  B2 * b2b2 = new B2;
  B2 * b2d21 = new D22;
  D21 * d21d22 = new D22;

  B1 * b1b1 = new B1;
  B1 * b1d2 = new D2;
  D1 * d1d2 = new D2;
  D2 * d2d2 = new D2;

  b2b2 -> f2(b1b1);
  b2b2 -> f2(b1d2);
  b2d21 -> f2(d1d2);
  b2d21 -> f2(d2d2);
  d21d22 -> f2(d1d2);
  d21d22 -> f2(d1d2);

  delete b2b2;
  delete b2d21;
  delete d21d22;
  delete b1b1;
  delete b1d2;
  delete d1d2;
  delete d2d2;

  return 0;
}
```

Answer:

```
B2:f2(B1 *) B1:f1(int)
B2:f2(B1 *) B1:f1(int)
D22:f2(B1 *) B1:f1(int)
D22:f2(D2 *) D2:f1(const char * )
```

```
D21:f2(D1 *) D1:f1(int)
D21:f2(D1 *) D1:f1(int)
```

5 Java Overloading and Overriding (Outcome 6, 3 points)

What is the output of this program?

```
class B1
{ public void f1() { System.out.println("B1:f1 "); } }
class D1 extends B1
{ public void f1() { System.out.println("D1:f1 "); } }
class D2 extends D1
{ public void f1() { System.out.println("D2:f1 "); } }
class B2
{
    public void f2(B1 b) { System.out.print("B2:f2(B1) "); b.f1(); }
    public void f2(D2 d) { System.out.print("B2:f2(D2) "); d.f1(); }
}
class D21 extends B2
{
    public void f2(D1 d) { System.out.print("D21:f2(D1) "); d.f1(); }
    public void f2(D2 d) { System.out.print("D21:f2(D2) "); d.f1(); }
}
class D22 extends D21
{
    public void f2(B1 d) { System.out.print("D22:f2(B1) "); d.f1(); }
    public void f2(D1 d) { System.out.print("D22:f2(D1) "); d.f1(); }
}
class Question
{
    public static void main(String [] args)
    {
        B2 b2b2 = new B2();
        B2 b2d22 = new D22();
        D21 d21d22 = new D22();
        B1 b1b1 = new B1();
        B1 b1d2 = new D2();
        D1 d1d2 = new D2();
        D2 d2d2 = new D2();
        b2b2.f2(b1b1);
        b2b2.f2(b1d2);
        b2d22.f2(d1d2);
        b2d22.f2(d2d2);
        d21d22.f2(d1d2);
        d21d22.f2(b1d2);
    }
}
```

Answer:

```
B2:f2(B1) B1:f1
B2:f2(B1) D2:f1
D22:f2(B1) D2:f1
D21:f2(D2) D2:f1
D22:f2(D1) D2:f1
D22:f2(B1) D2:f1
```

6 Group Programming Assignment (3 point)

What is the output of this command sequence?

```
RESET
U
R'
X
OUTPUT
```

1. RRORYGRYOWWWGRGGRGGOWYGWYOBBOBBOBYYYYBBYBWGBWOWRGWROOR
2. YYYYYYBOOGGYRRYRRYOGGOGGOGGWBBWOOWOORRRBBBBBBGRRWWWWWW
3. GYGRYYRBBYWWYRRRBBGRRYGGWRRYGWOOWYGOOBOOBOYBBOGGOWWWWB
4. YYBYYRGRRBBORROGOBYWWYGWRRWGGRGOOGOOYGOBBBBBYWYOGWWRWW
5. BYGBYGBRGRRRRRRWWWYGOYGWRGWYYYOOOOOORBWYBWYBOGOBGWBGWB
6. none of the above

Answer: 2