ECE 462 Exam 1

09:30-10:20AM, October 20, 2008

I certify that 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. Please put your photo ID on the desk.

This exam is printed **double sides**.

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

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

No electronic device is allowed. 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: an ability to overload operators in C++.
- Outcome 6: an ability to incorporate exception handling in object-oriented programs.

You need to obtain 50% of the points in each outcome to pass the outcome. There are 16 questions in this exam. If a program has a syntax error, **point out** which line causes the error and provide a **brief explanation**. If there are multiple errors, you need to write only one of them.

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".

Name:

Seat:

This page is blank.

Contents

1 C++ Exception Objects (outcome 6, 10 points) 4

2 Catch C++ Exception (outcome 6, 5 points) 5

3 Overload $\star$ Operator (outcome 5, 5 points) 6

4 Overload $-$ Operator (outcome 5, 5 points) 7

5 Overload $=$ Operator (outcome 5, 10 points) 8

6 Java Exception Class (outcome 6, 5 points) 9

7 Overload $<<$ Operator (outcome 5, 5 points) 10

8 UML Sequence Diagram (10 points) 11

9 Overload Postfix $++$ Operator (outcome 5, 5 points) 11

10 `friend` in C++ (outcome 5, 5 points) 13

11 Java Exception (outcome 6, 5 points) 13

12 Exception Handling (outcome 6, 5 points) 13

13 UML Class Diagram (10 points) 14

14 Java Exception and Array (outcome 6, 5 points) 15

15 Overload $=$, $<$, $>$ Operators (outcome 5, 5 points) 16

16 `try - catch` (outcome 6, 5 points) 16

Pass Outcomes:

Total Score:

1 C++ Exception Objects (outcome 6, 10 points)

What is the output of this program?
Answer:

```
caught Type 2 2
```

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

class ExceptionType1
{
protected:
  static int counter;
public:
  ExceptionType1() { counter ++; }
  virtual ~ExceptionType1() { counter --; }
  int getCounter()
  { return counter; }
};
int ExceptionType1::counter = 0;

class ExceptionType2: public ExceptionType1
{
public:
  ExceptionType2() { counter ++; }
};

void f (int i) throw (ExceptionType1, ExceptionType2)
{
  switch (i)
    {
    case 1:
      throw ExceptionType1();
      break;
    case 2:
      throw ExceptionType2();
      break;
    default:
      cout << "no exception" << endl;
    }
}

void g (int i) throw (ExceptionType1, ExceptionType2)
{
  try {
    f(i);
  } catch (ExceptionType1 et1) {
    throw ExceptionType2();
  }
}
```

```
int main()
{
  try {
    g(2);
    g(1);
    g(0);
  } catch (ExceptionType2 et2 ) {
    cout << "caught Type 2 " << et2.getCounter() << endl;
  } catch (ExceptionType1 et1 ) {
    cout << "caught Type 1 " << et1.getCounter() << endl;
  }
  return 0;
}
```

2 Catch C++ Exception (outcome 6, 5 points)

Please write the code so that the caller can `catch` the two types of exceptions.
Answer:

```
  } catch (ExceptionType1 et1 ) {
    cout << et1.getMessage() << endl;
  } catch (ExceptionType2 et2 ) {
    cout << et2.getValue() << endl;
  }
```

```
#include <iostream>
#include <string>
using namespace std;
class ExceptionType1
{
private:
  string et1_message;
public:
  ExceptionType1(string m): et1_message(m) { }
  string getMessage() const { return et1_message; }
};

class ExceptionType2
{
private:
  int et2_value;
public:
  ExceptionType2(int v) { et2_value = v; }
  int getValue() const { return et2_value; }
};

void f (int i) throw (ExceptionType1, ExceptionType2)
{
  switch (i)
    {
    case 1:
```

```
      throw ExceptionType1("type 1");
      break;
    case 2:
      throw ExceptionType2(2);
      break;
    default:
      cout << "no exception" << endl;
    }
}

int main()
{
  srand(time(NULL)); // initialize the random number
  try {
    f (rand() % 3);
  }
  // >>>>>
  // catch exception
  // if it is type 1, print the message
  // if it is type 2, print the value
  // <<<<<
  return 0;
}
```

3 Overload $\star$ Operator (outcome 5, 5 points)

Please overload the $\star$ operator between a Vector object and double.
Answer:

```
Vector operator * (const Vector & v, double d)
{
  return Vector (v.getX() * d, v.getY() * d, v.getZ() * d);
}
```

```
#include <iostream>
#include <string>
using namespace std;
// do not change the Vector class
class Vector
{
private:
  double v_x;
  double v_y;
  double v_z;
public:
  // do not change the public methods
  Vector(double x, double y, double z)
  { v_x = x; v_y = y; v_z = z; }
  void print()
  { cout << "(" << v_x << "," << v_y << "," << v_z << ")" << endl; }
  double getX() const { return v_x; }
```

```
  double getY() const { return v_y; }
  double getZ() const { return v_z; }
};
// >>>>>
// overload operator * for multiplication between a Vector object
// and double. return a Vector object
// <<<<<
int main(int argc, char * argv[])
{
  Vector v1(2, 3.1, 5.7);
  Vector v2 = v1 * 0.5;
  v1.print();
  v2.print();
  /* output:
     (2,3.1,5.7)
     (1,1.55,2.85)
  */
  return 0;
}
```

4 Overload – Operator (outcome 5, 5 points)

Please overload the – operator to change the direction of a vector object.
Answer:

```
Vector operator - ()
{
  return Vector (-v_x, -v_y, -v_z);
}
```

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

class Vector
{
private:
  double v_x;
  double v_y;
  double v_z;
public:
  Vector(double x, double y, double z)
  { v_x = x; v_y = y; v_z = z; }
  // >>>>>
  // overload - (negation) operator
  // <<<<<
  void print()
  { cout << "(" << v_x << "," << v_y << "," << v_z << ")" << endl; }
};
int main(int argc, char * argv[])
{
```

```
  Vector v1(2, 3.1, 5.7);
  Vector v2 = -v1;
  Vector v3(-0.8, 0.4, 7.5);
  v1.print();
  v2.print();
  v3.print();
  v3 = -v1;
  v3.print();
  /* output:
     (2,3.1,5.7)
     (-2,-3.1,-5.7)
     (-0.8,0.4,7.5)
     (-2,-3.1,-5.7)
  */
  return 0;
}
```

5 Overload = Operator (outcome 5, 10 points)

Please overload the = (assignment) operator for `Vector`.
Answer:

```
Vector & operator = (const Vector & v)
{
  if (this != & v)
    {
      delete [] v_element;
      v_size = v.v_size;
      v_element = new double[v_size];
      for (int ecnt = 0; ecnt < v_size; ecnt ++)
        { v_element[ecnt] = v.v_element[ecnt]; }
    }
  return * this;
}
```

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

class Vector
{
private:
  double * v_element;
  int v_size;
public:
  Vector(int s, double * elem)
  {
    v_size = s;
    v_element = new double[s];
    for (int ecnt = 0; ecnt < s; ecnt ++)
      { v_element[ecnt] = elem[ecnt]; }
```

```
  }
  // copy constructor
  Vector(const Vector & v)
  {
    v_size = v.v_size;
    v_element = new double[v_size];
    for (int ecnt = 0; ecnt < v_size; ecnt ++)
      { v_element[ecnt] = v.v_element[ecnt]; }
  }
  // >>>>>
  // overload = (assignment) operator
  // <<<<<
  void print()
  {
    for (int ecnt = 0; ecnt < v_size; ecnt ++)
      { cout << v_element[ecnt] << " "; }
    cout << endl;
  }
};
int main(int argc, char * argv[])
{
  double elem1[] = {1.1, 2.3, -5.3, 0.7, 9.2, 0.05, 3.3};
  double elem2[] = {2.1, 1.3, 4.3, 0.07, 5.2, 0.95, 2.3, 7.4, 8.3};
  int size1 = sizeof(elem1) / sizeof(double);
  int size2 = sizeof(elem2) / sizeof(double);
  Vector v1(size1, elem1);
  Vector v2 = v1;
  Vector v3(size2, elem2);
  v1 = v3;
  v1.print();
  v2.print();
  v3.print();
  /* output:
     2.1 1.3 4.3 0.07 5.2 0.95 2.3 7.4 8.3
     1.1 2.3 -5.3 0.7 9.2 0.05 3.3
     2.1 1.3 4.3 0.07 5.2 0.95 2.3 7.4 8.3
  */
  return 0;
}
```

6 Java Exception Class (outcome 6, 5 points)

Please write the class of `Exception641`.
Answer:

```
class Exception641 extends Exception {
    public Exception641( String s ) {
        super( s );
    }
}
```

```
class Exception641 // >>>>>
// fill the code for this class
// <<<<<
class outcome641 {
    static void f( ) throws Exception641 {
        throw new Exception641("thrown by f" );
    }
    public static void main( String[] args )
    {
        try {
            f();
        } catch( Exception641 exp ) {
            System.out.println( exp.getMessage() );
            // output:
            // thrown by f
        }
    }
}
```

7 Overload << Operator (outcome 5, 5 points)

Please overload the << operator to print the attributes of class `Person`.
Answer:

```
  friend ostream & operator << (ostream & os, const Person & p)
```

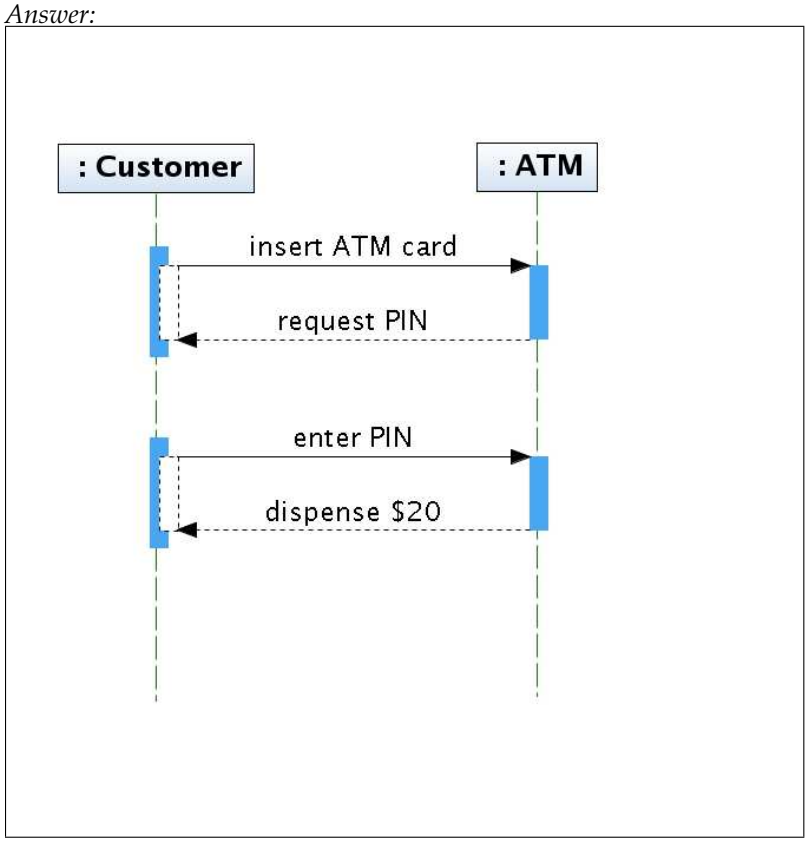
```
#include <iostream>
#include <string>
using namespace std;

class Person
{
private:
  string p_name;
  string p_address;
public:
  Person(string n, string a)
  { p_name = n; p_address = a; }
  // >>>>>
  // overload << operator
  // <<<<<
  {
    os << "name: " << p.p_name << endl;
    os << "address: " << p.p_address << endl << endl;
    return os;
  }
};
int main(int argc, char * argv[])
{
  Person p1("John", "123 Main Street");
  Person p2("Amy", "567 First Avenue");
```

```
  Person p3("Tom", "873 North Street");
  Person p4("Mary", "952 South Second Street");
  cout << p1;
  cout << p2;
  cout << p3;
  cout << p4;
  return 0;
}
```

8 UML Sequence Diagram (10 points)

Draw the sequence diagram to describe the following actions:

A customer inserts an ATM card into an ATM machine. The ATM machine asks the customer for PIN. The customer enters the PIN and then the ATM dispenses twenty dollars.
Answer:



9 Overload Postfix ++ Operator (outcome 5, 5 points)

Please overload the postfix ++ operator for `Vector`.
Answer:

```
const Vector Vector::operator ++ (int)
{
  Vector oldV = * this;
  v_x ++;
  v_y ++;
  v_z ++;
  return oldV;
}
```

```
#include <iostream>
#include <string>
using namespace std;
class Vector
{
private:
  int v_x;
  int v_y;
  int v_z;
public:
  Vector(int x, int y, int z)
  { v_x = x; v_y = y; v_z = z; }
  // >>>>>
  // postfix ++
  // <<<<<
  Vector & Vector::operator ++ ()   // prefix ++
  {
    v_x ++;
    v_y ++;
    v_z ++;
    return * this;
  }
  friend ostream & operator << (ostream & os, const Vector & v)
  {
    os << "(" << v.v_x << "," << v.v_y << "," << v.v_z << ")" << endl;
    return os;
  }
};
int main(int argc, char * argv[])
{
  Vector v1(2, 3, 5);
  cout << v1 << endl;
  cout << v1 ++ << endl;
  cout << ++ v1 << endl;
  /* output:
     (2,3,5)
     (2,3,5) <---- same as the previous one
     (4,5,7)
  */
  return 0;
}
```

10 `friend` in C++ (outcome 5, 5 points)

What does `friend` mean in C++?
Answer: *allows a class to access private attributes and methods*

11 Java Exception (outcome 6, 5 points)

What is the output of this program?
Answer:

```
0
1
2
3
Exception caught
```

```
import java.io.*;

class Err extends Exception { }
class outcome621 {
    public static void main( String[] args )
    {
        try {
            f(0);
        } catch( Err e ) {
            System.out.println( "Exception caught" );
        }
    }
    static void f(int j) throws Err {
        System.out.println( j );
        if (j == 3) throw new Err();
        f( ++j );
    }
}
```

12 Exception Handling (outcome 6, 5 points)

Which statement is correct?
Answer: *E*

A. In Java, a function may throw at most one type of exception.

B. In Java, an integer can be passed when an exception is thrown.

C. In C++, a function that may throw an exception must not call another function that may throw a different type of exception.

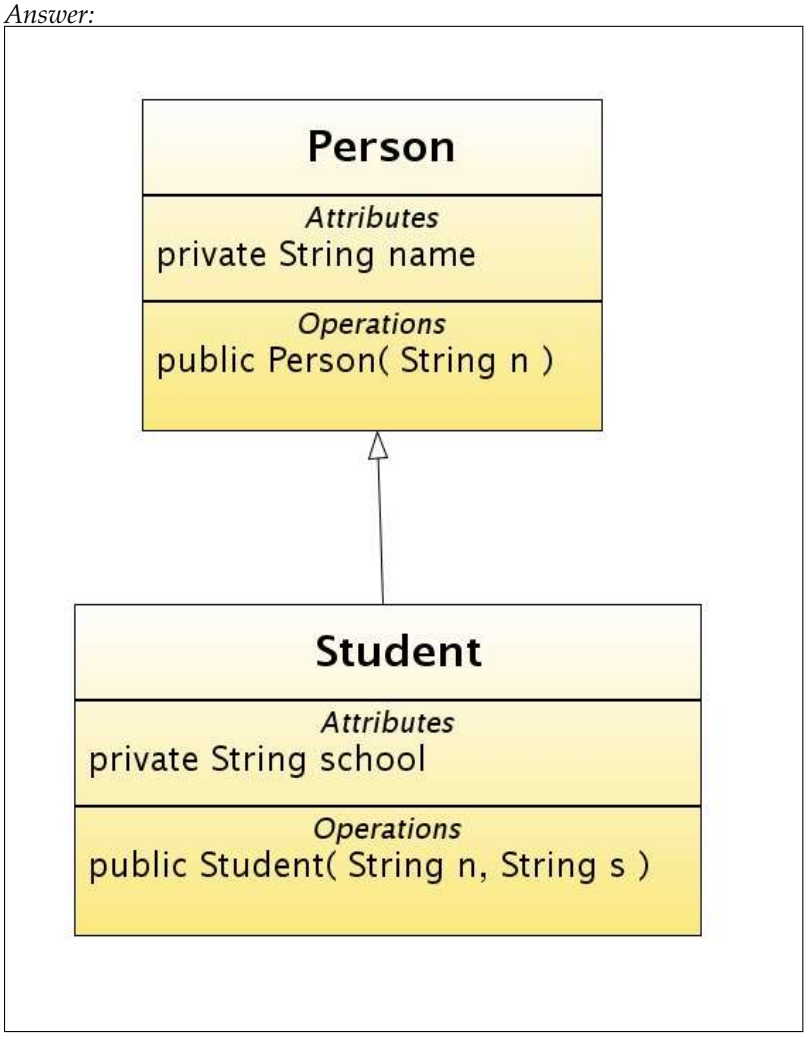
D. An exception must be caught by the immediate caller; otherwise the program terminates.

E. In Java, if an exception is not handled, the program terminates.

13 UML Class Diagram (10 points)

Draw the class diagram for the program.

Answer:




```
class Person
{
    private String name;
    public Person(String n)
    { name = n; }
}

class Student extends Person
{
    private String school;
    public Student(String n, String s)
    { super(n); school = s; }
}
```

14 Java Exception and Array (outcome 6, 5 points)

Please write the output of this program.
Answer:

```
0
1
2
3
4
caught exception
java.lang.ArrayIndexOutOfBoundsException: 5
program terminates
```

```
class Student
{
    public int s_ID;
    public Student(int id)
    {
        s_ID = id;
    }
    public String toString()
    {
        String str = String.valueOf(s_ID);
        return str;
    }
}

class outcome671
{
    public static void assignStudents(Student [] stus)
    {
        for (int scnt = 0; scnt < stus.length; scnt ++)
            { stus[scnt] = new Student(scnt); }
    }
    public static void main(String[] args)
    {
        try {
            int numStudents = 5;
            Student [] stus = new Student[numStudents];
            assignStudents(stus);
            // >>>>>
            // increase the value
            numStudents ++;
            // <<<<<
            for (int scnt = 0; scnt < numStudents; scnt ++)
                {
                    System.out.println(stus[scnt]);
                }
        } catch (Exception ept) {
            System.out.println("caught exception");
            System.out.println(ept);
```

```
        } finally {
            System.out.println("program terminates");
        }
    }
}
```

15 Overload $=$, $<$, $>$ Operators (outcome 5, 5 points)

What is the output of this program?
Answer:

```
no match for 'operator>' in 'p3 > p4'
```

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

class Person
{
private:
  string p_name;
  string p_address;
public:
  Person(string n, string a)
  { p_name = n; p_address = a; }
  bool operator < (const Person & p)
  { return (p_name < p.p_name); }
  bool operator == (const Person & p)
  { return (p_name == p.p_name); }
};
int main(int argc, char * argv[])
{
  Person p1("John", "123 Main Street");
  Person p2("Amy", "567 First Avenue");
  Person p3("Tom", "873 North Street");
  Person p4("Mary", "952 South Second Street");
  if ((p1 < p2) &&
      (p3 > p4))
    { cout << "team 1 won" << endl; }
  return 0;
}
```

16 `try - catch` (outcome 6, 5 points)

Which statement is correct?
Answer: *B*

A. In C++, the code inside `finally` is always executed regardless whether an exception has occurred.

B. The same function may throw different types of exceptions.

C. The code inside `catch` **cannot** throw any exception.

D. In Java, each `try` has one and only one correspoding `catch`.

E. If a function does **not** throw any exception, this function **cannot** be called inside a `try` block.