ECE 462 Exam 4

09:30-10:20AM, December 03, 2008

1 Timer using Thread (outcome 8)

Create a timer class so that it periodically calls the `update` function of an `Actuator` object.

Answer:

```
    public Outcome8Timer (Actuator a, int d, int r)
    {
        t_act = a;
        t_delay = d;
        t_repeat = r;
    }
    public void run()
    {
        for (int r = 0; r < t_repeat; r++)
            {
                try
                    {
                        t_act.update();
                        sleep(t_delay);
                    }
                catch (Exception ept)
                    {
                        System.out.println("caught exception");
                    }
            }
    }
```

```
class Actuator {
    public void update()
    {
        System.out.println("update called at " +
                           System.currentTimeMillis());
    }
}

class Outcome8Timer extends Thread {
    private Actuator t_act;
    private int t_delay;
    private int t_repeat;
    // >>>>>
    // The constructor initializes the three attributes
```

```
    // <<<<<




    // >>>>>
    // when start is called, call t_act.update() once every
    // t_delay millisecond.  This repeats t_repeat times.
    // If you use "sleep", please remember to enclose it inside
    // a try block.
    // <<<<<



}
public class outcome81 {
    public static void main( String[] args ) {
        Actuator act = new Actuator();
        Outcome8Timer t = new Outcome8Timer(act, 1500, 20);
        // call act.update every 1500 millisecond for 20 times
        t.start();
        try
            {
                t.join();
            }
        catch (Exception ept)
            {
                System.out.println("caught exception");
            }
    }
}
```

2 C++ Thread (outcome 8)

What are the differences of `wait` in C++/Qt and Java?

Answer:

In C++/Qt, `wait` *waits for a thread to complete the work in* `run`*. In Java,* `wait` *releases the lock in a synchronized method.*

3 Java Thread (outcome 8)

What are the **possible** outputs of the following program? If there are many possible outputs, describe their **common properties** (for example, positive integers between 50 and 200).

Answer:

integers around 100 but not always 100

```
class DataObject {
    int do_x1;
    int do_x2;
    DataObject() {
        do_x1 = 50;
        do_x2 = 50;
    }
    void swap() { // <--- not "synchronized"
        int x = (int) ( -4.999999 + Math.random() * 10 );
        do_x1 -= x;
        try { Thread.sleep( 1 ); } catch( InterruptedException e ) {}
        do_x2 += x;
    }
    void print() {
        int sum = do_x1 + do_x2;
        System.out.println( sum );
    }
}

class SwappingThread extends Thread  {
    DataObject dobj;

    SwappingThread(DataObject d) {
        dobj = d;
        start();
    }
    public void run( ) {
        int i = 0;
        while ( i++ < 20000 ) {
            dobj.swap();
            yield(); // Causes the currently executing thread object
                     // to temporarily pause and allow other threads
                     // to execute.
            if ( i % 40 == 0 ) dobj.print();
        }
    }
}

public class outcome83 {
    public static void main( String[] args ) {
        DataObject dobj = new DataObject();
        new SwappingThread(dobj);
        new SwappingThread(dobj);
        new SwappingThread(dobj);
        new SwappingThread(dobj);
    }
}
```

4 C++ Thread Conditions (outcome 8)

Please add code in appropriate locations so that the account balance is **never** negative. You need to modify **multiple** locations.

Answer:

```
class Account {
private:
  QMutex mutex;
  QWaitCondition cond;
  int balance;
public:
  Account() { balance = 0; }
  void deposit( int dep ) {
    mutex.lock();
    balance += dep;
    cond.wakeAll();
    mutex.unlock();
  }
  void withdraw( int draw ) {
    mutex.lock();
    while ( balance < draw ) {
      cond.wait( & mutex );
    }
    balance -= draw;
    mutex.unlock();
  }
  void getBalance() {
    mutex.lock();
    cout << "balance: " << balance << endl;
    mutex.unlock();
  }
};
```

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

class Account {
private:
  // >>>>>
  // add needed attributes
  // <<<<<



  int balance;
public:
  Account() { balance = 0; }
```

```
  void deposit( int dep ) {
    // >>>>>
    // add needed code
    // <<<<<


    balance += dep;



  }
  void withdraw( int draw ) {
    // >>>>>
    // add needed code
    // <<<<<



    while ( balance < draw ) {



    }



    balance -= draw;




  }
// ----------
// Do not modify anything below this line
// ----------
  void getBalance() {
    mutex.lock();
    cout << "balance: " << balance << endl;
    mutex.unlock();
  }
};
class Depositor : public QThread {
  Account * act;
public:
  Depositor (Account * a) { act = a; }
  void run() {
    int i = 0;
    while ( true ) {
      int x = (int) ( rand() % 10 );
      act -> deposit( x );
      if ( i++ % 100 == 0 )
```

```
        { act -> getBalance(); }
    }
  }
};
class Withdrawer : public QThread {
  Account * act;
public:
  Withdrawer(Account * a) { act = a; }
  void run() {
    int i = 0;
    while ( true ) {
      int x = (int) ( rand() % 100 );
      act -> withdraw( x );
      if ( i++ % 100 == 0 )
        { act -> getBalance(); }
    }
  }
};
int main()
{
  Account act;
  Depositor* depositors[5];
  Withdrawer* withdrawers[5];

  for ( int i=0; i < 5; i++ ) {
    depositors[ i ] = new Depositor(& act);
    withdrawers[ i ] = new Withdrawer(& act);
    depositors[ i ]->start();
    withdrawers[ i ]->start();
  }
  for ( int i=0; i < 5; i++ ) {
    depositors[ i ]->wait();
    withdrawers[ i ]->wait();
  }
}
```

5 Java Synchronized Methods (outcome 8)

Briefly explain whether a deadlock is **possible** in this program.

Answer:

no deadlock because there is no hold and wait.

```
class Account {
    int a_id;
    Account(int i) { a_id = i; }
    synchronized void func1 () { func2(); }
    synchronized void func2 () { }
```

```
}

class ActionThread extends Thread {
    Account tt_act1;
    Account tt_act2;
    ActionThread(Account a1, Account a2) {
        tt_act1 = a1;
        tt_act2 = a2;
    }
    public void run() {
        tt_act1.func1();
        tt_act2.func1();
    }
}

public class outcome85 {
    public static void main( String[] args ) {
        Account act1 = new Account(3);
        Account act2 = new Account(19);
        ActionThread tt1 = new ActionThread(act1, act2);
        ActionThread tt2 = new ActionThread(act1, act2);
        tt1.start();
        tt2.start();
        try {
            tt1.join();
            tt2.join();
        } catch (Exception je) {
            System.out.println("join exception " + je);
        }
        System.out.println("all funcactions completed");
    }
}
```

6 Java Producer - Consumer (outcome 8)

In the following producer - consumer program, which functions **must** be `synchronized` so that (1) every produced item is consumed exactly once, (2) the elements are removed in the order as they are added (first-in-first-out), and (3) only produced items are consumed. The shared buffer does neither overflow nor underflow.

Mark only the methods that **must** *be* `synchronized`*. You will lose 2 points for each method that does not have to be* `synchronized`*.*

Answer:

`SharedBuffer.put` *and* `SharedBuffer.get`

```
class SharedBuffer
{
```

```
    private int sb_head;
    private int sb_tail;
    private int sb_numElem; // number of valid elements
    private int [] sb_element; // circular buffer
    public SharedBuffer(int size)
    {
        sb_element = new int [size];
        sb_head = 0;
        sb_tail = 0;
        sb_numElem = 0;
    }
    public void put(int v)
    {
        while (sb_numElem >= sb_element.length)
            {
                try
                    { wait(); }
                catch (InterruptedException e )
                    { System.out.println("caught exception in put"); }
            }
        sb_numElem ++;
        sb_element [sb_tail] = v;
        sb_tail = (sb_tail + 1) % sb_element.length;
        notifyAll();
    }
    public int get()
    {
        while (sb_numElem < 0)
            {
                try
                    { wait(); }
                catch (InterruptedException e )
                    { System.out.println("caught exception in get"); }
            }
        int v = sb_element [sb_head];
        sb_numElem --;
        sb_head = (sb_head + 1) % sb_element.length;
        notifyAll();
        return v;
    }
}
class Producer extends Thread
{
    private SharedBuffer p_sbuf;
    private final int p_numItem = 300;
    public Producer(SharedBuffer buf)
    { p_sbuf = buf; }
    public void run()
    {
        for (int iter = 0; iter < p_numItem; iter ++)
            { p_sbuf.put(iter); }
```

```
    }
}
class Consumer extends Thread
{
    private SharedBuffer c_sbuf;
    private final int c_numItem = 300;
    public Consumer(SharedBuffer buf)
    { c_sbuf = buf; }
    public void run()
    {
        int val;
        for (int iter = 0; iter < c_numItem; iter ++)
            { val = c_sbuf.get(); System.out.println(val); }
    }
}
public class outcome881 {
    public static void main (String[] args ) {
        // create a shared buffer
        int bufSize = 20;
        SharedBuffer sbuf = new SharedBuffer(bufSize);
        // create threads
        int numThread = 4;
        Producer [] prod = new Producer [numThread];
        Consumer [] cons = new Consumer [numThread];
        for (int index = 0; index < numThread; index ++)
            {
                prod[index] = new Producer (sbuf);
                cons[index] = new Consumer (sbuf);
            }
        // start producing and consuming
        for (int index = 0; index < numThread; index ++)
            {
                prod[index].start();
                cons[index].start();
            }
        // wait until all threads complete
        try
            {
                for (int index = 0; index < numThread; index ++)
                    {
                        prod[index].join();
                        cons[index].join();
                    }
            }
        catch (Exception je)
            { System.out.println("join exception " + je); }
    }
}
```

7 Java Polymorphism (Outcome 3)

What is the output of this program?

Answer:

```
name: Charlie
school: Lafayette
```

```
import java.io.*;
import java.util.*;

class Person
{
    private String p_name;
    public Person(String n) {  p_name = n; }
    public void print()
    { System.out.println("name: " + p_name); }
}
class Student extends Person
{
    private String s_school;
    public Student(String n, String sch)
    {
        super(n);
        s_school = sch;
    }
    public void print()
    {
        super.print();
        System.out.println("school: " + s_school);
    }
}
class outcome31
{
    public static void main(String[] args)
    {
        Person [] per = new Person[2];
        per[0] = new Person("John");
        per[1] = new Student("Amy", "Purdue");
        Student [] stu = new Student[2];
        stu[0] = new Student("Jennifer", "Indiana");
        stu[1] = new Student("Charlie", "Lafayette");
        per[1] = stu[1];
        per[1]. print();
    }
}
```

8 Java Derived Class (Outcome 3)

What is the output of this program?

Answer:

7

```
import java.io.*;
import java.util.*;

class Person
{
    private String p_name;
    protected int p_val;
    public Person(String n) { p_val = 0; p_name = n; }
    public void func1() { p_val ++; }
    public void func2() { p_val += 2; }
    public int getVal() { return p_val; }
}
class Student extends Person
{
    private String s_school;
    public Student(String n, String sch)
    {
        super(n);
        s_school = sch;
    }
    public void func1() { p_val += 3; }
}
class CollegeStudent extends Student
{
    private String cs_major;
    public CollegeStudent(String n, String sch, String ma)
    {
        super(n, sch);
        cs_major = ma;
    }
    public void func2() { p_val += 4; }
};
class outcome32
{
    public static void main(String[] args)
    {
        Person [] per = new Person[3];
        per[0] = new Person("John");
        per[1] = new Student("Amy", "Purdue");
        per[2] = new CollegeStudent("Tom", "West Lafayette", "ECE");
        Student [] stu = new Student[2];
        stu[0] = new Student("Jennifer", "Indiana");
        stu[1] = new CollegeStudent("Mary", "Indiana", "Math");
```

```
        stu[1].func1();
        stu[1].func2();
        System.out.println(stu[1].getVal());
    }
}
```

9 C++ Abstract Class (Outcome 3)

How to make `Shape` an abstract class and `Triangle` must override the `getArea` method?

Answer:

```
virtual double getArea() = 0;
```

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

class Shape
{
public:
  Shape() { }
  // >>>>>
  // getArea must be overridden in derived classes
  // <<<<<
};

class Triangle: public Shape
{
public:
  Triangle(double h, double w)
  {
    width = w;
    height = h;
  }
  double getArea()
  {
    return (0.5 * height * width);
  }
private:
  double height;
  double width;
};

int main(int argc, char * argv[])
{
  Shape * sobj1 = new Triangle(4.0, 5.0);
  cout << sobj1 -> getArea() << endl;
  return 0;
}
```

10 Java Classes (Outcome 3)

What is the output of this program?

Answer:

```
name: John
name: Amy
school: Purdue
```

```
import java.io.*;
import java.util.*;

class Person
{
    private String p_name;
    public Person(String n) {  p_name = n; }
    public void print()
    { System.out.println("name: " + p_name); }
}

class Student extends Person
{
    private String s_school;
    public Student(String n, String sch)
    {
        super(n);
        s_school = sch;
    }
    public void print()
    {
        super.print();
        System.out.println("school: " + s_school);
    }
}
class outcome34
{
    public static void main(String[] args)
    {
        final int numPerson = 2;
        Person [] per = new Person[numPerson];
        per[0] = new Person("John");
        per[1] = new Student("Amy", "Purdue");
        for (int scnt = 0; scnt < numPerson; scnt ++)
            { per[scnt].print();  }
    }
}
```

11 Java Inheritance (Outcome 3)

In Java, if class X is the immediate **base** class of Y, how to express this relation?

Answer: `class Y extends X`

12 C++ Constructor and Destructor (Outcome 3)

What is the output of this program? (two answers)

Answer: *7 2*

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

class Person
{
public:
  static int counter; // <--- static
  Person()  { counter ++; }
  ~Person()  { counter --; } // <--- not virtual
};

class Student: public Person
{
public:
  Student() { counter ++; }
  ~ Student() { counter --; } // <--- not virtual
};

int Person::counter = 0;

int main(int argc, char * argv[])
{
  const int numPerson = 2;
  Person * per[numPerson];
  per[0] = new Person;
  per[1] = new Student;
  Person * p1 = new Student;
  Student * s1 = new Student;
  cout << Person::counter << endl;
  delete p1;
  delete s1;
  delete per[0];
  delete per[1];
  cout << Person::counter << endl;
  return 0;
}
```

13 Java Vector (Outcome 4)

How to use the iterator in function `print`?

Answer:

```
while (p.hasNext())
```

```
import java.io.*;
import java.util.*;
class outcome41
{
    public static void print(Vector<String> vec)
    {
        System.out.println("vector size is: " + vec.size());
        Iterator<String> p = vec.iterator();
        // >>>>>
        // print all elements in vec
        // <<<<<
            { System.out.println(p.next()); }
    }
    public static void main(String[] args)
    {
        Vector<String> vec = new Vector<String>();
        vec.add("hello");
        vec.add("ece462");
        vec.add("C++");
        vec.add("Java");
        print(vec);
    }
}
```

14 Java List (Outcome 4)

What is the output of this program?

Answer:

```
list size is: 2
c++
java
```

```
import java.io.*;
import java.util.*;
class outcome42
{
    public static void print(List<String> lis)
    {
```

```
        System.out.println("list size is: " + lis.size());
        Iterator p = lis.iterator();
        while (p.hasNext())
            { System.out.println(p.next()); }
    }
    public static void main(String[] args)
    {
        List<String> ece462 = new ArrayList<String>();
        ece462.add("c++");
        ece462.add("fall");
        ece462.add("2008");
        ece462.remove("fall");
        ece462.add("java");
        ece462.remove("2008");
        ece462.remove("2008"); // <--- remove again
        print(ece462);
    }
}
```

15 C++ Queue (Outcome 4)

What is the output of this program?

Answer:

```
Amy
```

```
#include <iostream>
#include <string>
#include <queue>
using namespace std;
class Student
{
public:
  string s_name;
  int s_id;
  Student(int id, string name)
  { s_id = id; s_name = name; }
  void print()
  { cout << s_name << " " << s_id << endl; }
};

int main(int argc, char * argv[])
{
  queue<Student> qs;
  Student s1(12345, "Amy");
  Student s2(65412, "John");
  qs.push(s1);
  qs.push(s2);
```

```
  s1.s_id = 98741;
  s2.s_id = 462;
  s1.s_name = "Tom";
  s2.s_name = "Mary";
  cout << (qs.front()).s_name << endl; // Amy? Tom? Mary? John?
  return 0;
}
```

16 C++ Template (Outcome 4)

Write the function or `insert`.

Answer:

```
    ECE462List<TPL> * next = new ECE462List<TPL>(content);
    next -> el_next = el_next;
    el_next = next;
```

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

class Student
{
public:
  int s_ID;
  string s_name;
  Student(int id, string name): s_ID(id), s_name(name)  { }
  void print()
  { cout << "ID: " << s_ID << " name: " << s_name << endl; }
};

template <class TPL> class ECE462List
{
private:
  TPL * el_content;
  ECE462List * el_next;
public:
  ECE462List(TPL * content): el_content(content)  { el_next = 0; }
  void insert(TPL * content)
  {
    // <<<
    // write the insert function
    // >>>
  }
  void print()
  {
```

```
    el_content -> print();
    ECE462List * node = el_next;
    while (node != 0)
      {
        (node -> el_content) -> print();
        node = node -> el_next;
      }
    cout << endl << endl;
  }
  ~ ECE462List()
  {
    // ignore destructor in this exam
  }
};

int main(int argc, char * argv[])
{
  Student * s1 = new Student (12345, "Amy");
  Student * s2 = new Student (65412, "John");
  Student * s3 = new Student (25789, "Tom");
  Student * s4 = new Student (78941, "Jennifer");
  ECE462List<Student> * head = new ECE462List<Student>(s1);
  head -> insert(s2);
  head -> insert(s3);
  head -> insert(s4);
  head -> print();
  delete head;
  delete s1;
  delete s2;
  delete s3;
  delete s4;
  return 0;
}
```

17 Java Set (Outcome 4)

What is the output of this program?

Answer:

`[John 2, John 2, Amy 1, Amy 1]` *(Order does not matter.)*

```
import java.io.*;
import java.util.*;

class Student
{
    public String s_name;
    public int s_ID;
```

```
    public Student(String name, int id)
    {
        s_name = name;
        s_ID = id;
    }
    public String toString()
    {
        String str = s_name + " " + String.valueOf(s_ID);
        return str;
    }
}
class outcome45
{
    public static void main(String[] args)
    {
        Set<Student> ecestudent = new HashSet<Student>();
        Student s1 = new Student("Amy", 1);
        Student s2 = new Student("John", 2);
        Student s3 = new Student("Amy", 1);
        Student s4 = new Student("John", 2);
        ecestudent.add(s1);
        ecestudent.add(s2);
        ecestudent.add(s3);
        ecestudent.add(s4);
        System.out.println(ecestudent);
    }
}
```

18 C++ Set (Outcome 4)

Implement a C++ set using list. For simplicity, this queue can handle only strings. Please remember that a set has **no** duplicate element.

Answer:

```
  void add(string str)
  {
    list<string>::iterator iter = s_list.begin();
    while (iter != s_list.end())
      {
        if ((*iter) == str) // duplicate
          {
            return;
          }
        iter ++;
      }
    s_list.push_back(str);
  }
  bool remove(string str)
  {
    list<string>::iterator iter = s_list.begin();
    while (iter != s_list.end())
      {
        if ((*iter) == str)
          {
            s_list.remove(str);
            return true;
          }
        iter ++;
      }
    return false;
  }
```

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

class ECE462Set
{
private:
  list<string> s_list;
public:
  ECE462Set() { }
  void add(string str)
  {
    // >>>>>
    // add the string, must not have duplicate elements, use iterator
    // to check whether the element is already in the set
    // <<<<<



  }
  bool remove(string str)
  {
```

```
    // >>>>>
    // remove the string
    // return true if str is originally in the set
    // return false if str is not originally in the set
    // <<<<<



  }
};
int main(int argc, char * argv[])
{
  ECE462Set eset;
  eset.add("nice");
  eset.add("day");
  eset.add("nice");
  eset.add("day");
  cout << eset.remove("nice") << endl;
  cout << eset.remove("day") << endl;
  cout << eset.remove("nice") << endl;
  cout << eset.remove("day") << endl;
  return 0;
}
```

19 Overload << Operator (outcome 5)

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;
  /* output:
  name: John
  address: 123 Main Street
  name: Amy
  address: 567 First Avenue
  name: Tom
  address: 873 North Street
  name: Mary
  address: 952 South Second Street
  */
  return 0;
}
```

20 Overload – Operator (outcome 5)

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;
}
```

21 Overload * Operator (outcome 5)

Please overload the * 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;
}
```

22 Overload $=$, $<$, $>$ Operators (outcome 5)

What is the output of this program?

Answer:

```
team 2 won
```

```
#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); }
  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; }
  else
    { cout << "team 2 won" << endl; }
  return 0;
}
```

23 Overload = Operator (outcome 5)

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;
}
```

24 Overload Prefix ++ Operator (outcome 5)

Please overload the prefix ++ operator for `Vector`.

Answer:

```
Vector & Vector::operator ++ ()   // prefix ++
{
  v_x ++;
  v_y ++;
  v_z ++;
  return * this;
}
```

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

25 Java Exception (outcome 6)

What is the output of this program?

Answer:

```
-3
No exception
```

```
import java.io.*;

class Err extends Exception { }
class outcome61 {
    public static void main( String[] args )
    {
        try {
            f( -3 );
        } catch( Err e ) {
            System.out.println("Exception caught" );
        } finally {
            System.out.println("No exception");
        }
    }
    static void f(int j) throws Err {
        System.out.println(j);
        if (j >= 3) { throw new Err(); }
        if (j < 0)  { return; }
        f( ++j );
    }
}
```

26 Java Exception Class (outcome 6)

Through an exception with an object of `Exception62` and a message "Java Exception".

Answer:

```
        throw new Exception62("Java Exception" );
```

```
class Exception62 extends Exception {
    public Exception62(String s) { super( s ); }
}
class outcome62 {
    static void f( ) throws Exception62 {
        // >>>>>
        // throw an exception of Exception62 with
        // "Java Exception" as the message
        // <<<<<
    }
    public static void main( String[] args )
    {
        try {
            f();
        } catch( Exception62 exp ) {
            System.out.println( exp.getMessage() );
            // output:
            // Java Exception
        }
    }
}
```

27 Catch C++ Exception (outcome 6)

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;
}
```

28 C++ Exception Objects (outcome 6)

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

```
  return 0;
}
```

29 C++ Exception and Array (outcome 6)

Please write the output of this program. This question asks you whether C++ checks array indexes and throws exceptions.

Answer:

some random values or crash. C++ does not check array indexes.

```
#include <iostream>
using namespace std;
class Student
{
public:
  Student(int id = 0)  { s_ID = id; }
  void print()  { cout << "ID " << s_ID << endl; }
private:
  int s_ID;
};

int main(int argc, char * argv[])
{
  Student st[2];
  try
    {
      for (int scnt = -1; scnt < 100; scnt ++) // <--- range?
        { st[scnt].print(); }
    }
  catch (int x)
    { cout << "caught exception" << endl; }
  return 0;
}
```

30 Exception Execution Order (outcome 6)

Please write the output of this program.

Answer:

```
11 = 2 * 3 + 5
```

```
class Person
{
    public int p_ID;
    public Person() { p_ID = 2; }
    public void func1() { p_ID *= 3; }
    public void func2() { p_ID += 5; }
}

class outcome66
{
    public static void func3(Person p, boolean v) throws Exception
    {
        if (v == true) { throw new Exception("by func3"); }
        p.func1();
        p.func2();
    }
    public static void main(String[] args)
    {
        Person p = new Person();
        try {
            func3(p, false);
        } catch (Exception ept) {
            p.func2();
        } finally {
            System.out.println(p.p_ID);
        }
    }
}
```

31 Virtual and Overloaded Function in C++ (outcome 7)

What is the output of this program?

Answer:

8 = 3 Derived1::f1 + 5 Derived2::f2

```
#include <iostream>
using namespace std;
int gvalue = 0;
class Base
{
public:
  virtual void f1() { gvalue += 1; }
  virtual void f2() { gvalue += 2; }
};

class Derived1: public Base
{
```

```
public:
  void f1() { gvalue += 3; } // <--- not virtual
  void f2() { gvalue += 4; }
};

class Derived2: public Derived1
{
public:
  void f1(int v) { gvalue += v; }
  void f2() { gvalue += 5; }
};

int main(int argc, char * argv[])
{
  Base * bobj = new Derived2;
  bobj -> f1();
  bobj -> f2();
  cout << gvalue << endl;
  return 0;
}
```

32 Overload and Override in Java (outcome 7)

What is the output of this program?

Answer:

*20 = 1 + 2 * 7 + 5*

```
class Base
{
    public static int gvalue = 1;
    public void f1() { gvalue += 1; }
    public void f2() { gvalue += 2; }
}

class Derived1 extends Base
{
    public void f1(int v) { gvalue += v; }
    public void f2() { gvalue += 4; }
}

class Derived2 extends Derived1
{
    public void f1(int v) { gvalue += 2 * v; }
    public void f2() { gvalue += 5; }
}

class outcome72 {
```

```
    public static void main( String[] args ) {
        Derived1 dobj = new Derived2();
        dobj.f1(7);
        dobj.f2();
        System.out.println(dobj.gvalue);
    }
}
```

33 Overload using Return Types (outcome 7)

What is the output of this program?

Answer:

```
outcome73.java:9: f1() in Derived1 cannot override f1() in Base;
attempting to use incompatible return type
found   : void
required: java.lang.String
    public void f1() { }
                ^
outcome73.java:10: f2() in Derived1 cannot override f2() in Base;
attempting to use incompatible return type
found   : int
required: java.lang.String
    public int f2() { return 4; }
                ^
outcome73.java:15: f1() in Derived2 cannot override f1() in Base;
attempting to use incompatible return type
found   : void
required: java.lang.String
    public void f1() { System.out.println("Derived2.f1"); }
                ^
outcome73.java:16: f2() in Derived2 cannot override f2() in Derived1;
attempting to use incompatible return type
found   : double
required: int
    public double f2() { return 4.62; }
                  ^
outcome73.java:22: 'void' type not allowed here
        System.out.println(dobj.f1());
                                   ^
5 errors
```

```
class Base
{
    public String f1() { return "Base.f1"; }
    public String f2() { return "Base.f2"; }
}
```

```
class Derived1 extends Base
{
    public void f1() { }
    public int f2() { return 4; }
}

class Derived2 extends Derived1
{
    public void f1() { System.out.println("Derived2.f1"); }
    public double f2() { return 4.62; }
}

class outcome73 {
    public static void main( String[] args ) {
        Derived2 dobj = new Derived2();
        System.out.println(dobj.f1());
        System.out.println(dobj.f2());
    }
}
```

34 Overloading with Type Conversion (outcome 7)

What is the output of this program?

Answer:

```
Bd
D1s
```

```
class Base
{
    public void f1(double v) { System.out.println("Bd"); }
}

class Derived1 extends Base
{
    public void f1(String v) { System.out.println("D1s"); }
    public void f1(int v)    { System.out.println("D1i"); }
}

class Derived2 extends Derived1
{
    public void f1(double v) { System.out.println("D2d"); }
}

class outcome74 {
    public static void main( String[] args ) {
```

```
        Base bd1obj = new Derived1();
        bd1obj.f1(3.14159);
        Derived1 d1d2obj = new Derived2();
        d1d2obj.f1("ece462");
    }
}
```

35 Virtual Function in C++ (outcome 7)

What is the output of this program?

Answer:

```
B()
D2(int)
```

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

class Base
{
public:
  void f1()                 { cout << "B()" << endl; } // <--- not virtual
  void f1(string s)         { cout << "B(string)" << endl; }
  virtual void f1(int v)    { cout << "B(int)" << endl; }
  virtual void f1(double v) { cout << "B(double)" << endl; }
};

class Derived1: public Base
{
public:
  void f1()                 { cout << "D1()" << endl; }
  virtual void f1(int v)    { cout << "D1(int)" << endl; }
};

class Derived2: public Derived1
{
public:
  void f1(string s)         { cout << "D2(string)" << endl; }
  virtual void f1(int v)    { cout << "D2(int)" << endl; }
};

int main(int argc, char * argv[])
{

  Base * obb = new Derived1;
  Base * obd1 = new Derived1;
  Derived1 * od1d2 = new Derived2;
```

```
  obb  -> f1();
  od1d2 -> f1(462);
  return 0;
}
```

36 Overloading in Java (outcome 7)

What is the output of this program?

Answer:

```
416 = 64 + 128 + 64 + 128 + 32
```

```
class Base
{
    protected static int val = 0; // <--- static
    public void f1()         { val += 1; }
    public void f1(int v)    { val += 2; }
    public void f1(double v) { val += 4; }
    public void f1(String s) { val += 8; }
    public int getVal() { return val; }
}

class Derived1 extends Base
{
    public void f1(int v)    { val += 16; }
    public void f1(double v) { f1("fall-2008"); val += 32; }
}

class Derived2 extends Derived1
{
    public void f1()         { val += 64; }
    public void f1(String s) { f1(); val += 128; }
}

class outcome76 {
    public static void main( String[] args ) {
        Base obd2 = new Derived2();
        obd2.f1("ece462");
        obd2.f1(12.9);
        System.out.println(obd2.getVal());
    }
}
```