# Inf1 OOP Resit Exam: Sample Solutions

## Question 1 Solutions

### `OneA.java`

```
public class OneA {

        public static double geometricMean(int[] nums) {

                int product = 1;
                for (int n : nums) {
                        product *= n;
                }
                return Math.pow(product, (1.0 / nums.length));
        }

}
```

### `OneB.java`

```
public class OneB {

        public static int longestSeq(int[] nums, int n) {
                int max = 0;
                int curLength = 0;
                for (int i : nums) {
                        if (i == n) {
                                curLength++;
                                if (curLength > max)
                                        max = curLength;
                        } else
                                curLength = 0;
                }
                return max;
        }

}
```

## OneC.java

```
import java.util.ArrayList;
import java.util.HashMap;

public class OneC {

    public static ArrayList<String> findSubstrings(String s, int len) {
         ArrayList<String> substrings = new ArrayList<String>();
         for (int i = 0; i < s.length() - (len - 1); i++) {
             substrings.add(s.substring(i, i + len));
         }
         return substrings;
     }


    public static void increment(HashMap<String, Integer> map, String s) {
        Integer count = map.get(s);
        if (count == null)
            map.put(s, 1);
        else
            map.put(s, count + 1);
    }

    public static HashMap<String, Integer> findStringCounts(String s) {
        HashMap<String, Integer> freq = new HashMap<String, Integer>();
        ArrayList<String> substrings = findSubstrings(s, 3);
        for (String sub : substrings) {
            increment(freq, sub);
        }
        return freq;
    }
}
```

## OneD.java

No sample answer for this question.

# Question 2 Solutions

## `Element.java`

```
public class Element {

    private final String element;
    private final int atomicNum;
    private final String sym;
    private final double weight;

    public Element(String element, int atomicNum, String sym, double weight) {
        this.element = element;
        this.atomicNum = atomicNum;
        this.sym = sym;
        this.weight = weight;
    }

    public String getElement() {
        return element;
    }

    public int getAtomicNum() {
        return atomicNum;
    }

    public String getSym() {
        return sym;
    }

    public double getWeight() {
        return weight;
    }

    public String toString() {
        String val = String.format("Element(%s, %s, %s, %.2f)",
                                   element, atomicNum, sym, weight);
        return val;
    }
}
```

## Table.java

```
import java.util.ArrayList;

public class Table {

        private static boolean lookupCalled;
        private static String lookupParam;

        public static boolean lookupCalled() {
                return lookupCalled;
        }

        public static String lookupParam() {
                return lookupParam;
        }

        private ArrayList<Element> table = new ArrayList<Element>();

        public Table(String fn) {
                readFile(fn);
        }

        public Table() {
                this("elements.csv");
        }

        public void readFile(String fn) {
                In file = new In(fn);
                while (!file.isEmpty()) {
                        String line = file.readLine();
                        String[] parts = line.split(",");
                        Element e = new Element(parts[0], Integer.parseInt(parts[1]),
                                        parts[2], Double.parseDouble(parts[3]));
                        table.add(e);
                }
        }

        public Element lookup(String sym) {
                lookupCalled = true;
                lookupParam = sym;
                Element result = null;
                for (Element e : table) {
                        if (e.getSym().equals(sym))
                                result = e;
                }
                return result;
        }

        public void display() {
                for (Element e : table) {
                        System.out.println(e);
                }
        }


}
```

## `Table.java` using a HashMap

```
import java.util.ArrayList;
import java.util.HashMap;

public class Table {

    private HashMap<String, Element> map = new HashMap<String, Element>();
    private ArrayList<Element> table = new ArrayList<Element>();

    public Table(String fn) {
        readFile(fn);
    }

    public Table(){
        this("elements.csv");
    }

    public void readFile(String fn) {
        In file = new In(fn);
        while (!file.isEmpty()) {
            String[] parts = file.readLine().split(",");
            String elementName = parts[0];
            int atomicNum = Integer.parseInt(parts[1]);
            String sym = parts[2];
            double weight = Double.parseDouble(parts[3]);
            table.add(new Element(elementName, atomicNum, sym, weight));
        }
        for (Element e : table) {
            map.put(e.getSym(), e);
        }
    }

    public Element lookup(String sym) {
        return map.get(sym);
    }

    public void display() {
        for (Element e : table)
            System.out.println(e);
    }

}
```

## Molecule.java

```
import java.util.ArrayList;
import java.util.HashMap;

public class Molecule {

    private Table pt = new Table();
    private HashMap<Element, Integer> structure = new HashMap<Element, Integer>();

     public void addAtom(String sym, int num) {
         Element e = t.lookup(sym);
         structure.put(e, num);
     }

     public void addAtom(String sym) {
         addAtom(sym, 1);
     }


    public ArrayList<Element> atoms() {
         ArrayList<Element> atoms = new ArrayList<Element>();
         for (Element e : structure.keySet()) {
             atoms.add(e);
         }
         return atoms;
     }

    public double weight() {
         double weight = 0;
         for (Element e : atoms()) {
             weight += e.getWeight() * structure.get(e);
         }
         return weight;
     }

}
```