1. (3 points): Suppose we’d like to be able to use Gorp objects directly in a foreach loop, like this:
   ```java
   for (Integer g: aGorp)
   ...
   ```
   To do so, what interface would Gorp have to implement:
   (a) Container
   (b) Iterable
   (c) Iterator
   (d) Object

2. (3 points): Any class implementing the Iterator interface must provide what methods?
   (a) enum(), run(), add()
   (b) hasNext(), next(), remove()
   (c) iterator()
   (d) toString(), hash(), and a constructor

3. (3 points): Suppose a class has two methods with the same name. Code using this class would indicate a choice between these methods by
   (a) flipping a coin.
   (b) the names of the parameters.
   (c) the precise spelling of the method names.
   (d) the types and number of parameters.
4. **(10 points):** Consider the following class:

```java
private class DeckIterator implements Iterator<Card> {
    private int current;

    public DeckIterator() {
        current = 0;
    } // no-arg constructor

    public boolean hasNext() {
        return current < n;
    } // hasNext()

    public Card next() {
        return theDeck[current++];
    } // next()

    public void remove() {
        final String s =
            "DeckIterator.remove() not implemented."
            throw new UnsupportedOperationException(s);
    } // remove()

} // class DeckIterator
```

Suppose this is used to deal cards for a card game. Further suppose that we want the dealer to be able to peek at the top card of the deck. Write a method called `peek()` that could be added to `DeckIterator` to return the top card on the deck without advancing the iterator to the next card.
5. (10 points): Briefly describe the output of the following program:

```java
import java.util.Iterator;
public class T2It implements Iterable<Integer> {
    private int startingPoint;
    public T2It(int start) {
        startingPoint = start;
    } // constructor

    public Iterator<Integer> iterator() {
        return new T2Iterator(startingPoint);
    } // iterator()

    private class T2Iterator implements Iterator<Integer> {
        private int current;
        public T2Iterator(int start) {
            current = start;
        } // no-arg constructor
        public boolean hasNext() {
            return true;
        } // hasNext()
        public Integer next() {
            return new Integer(current++);
        } // next()
        public void remove() {
        } // remove()
    } // class T2Iterator

    public static void main(String[] args) {
        T2It t = new T2It(3);
        for (Integer val : t)
            System.out.println(val);
    } // main()
} // class T2It
```
6. (10 points): Consider the following Java class, where the ellipses (...) indicate code that exists but isn’t shown:

```java
public class T2Root {
    private double number;

    // ...

    public double root() {
        return Math.sqrt(number);
    }
    //

    // ...

} // class T2Root
```

Write a complete class that behaves exactly the same as class T2Root except that in your class root() returns the fourth root of number rather than the square root. Recall that the fourth root is the square root of the square root. Since most of the code from T2Root has been hidden from you, the only way to do this is through inheritance.
7. **(10 points):** Show the output produced by the following program:

```java
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class T2Last {

    public static List<String> scannerToList(Scanner in) {
        List<String> theList = new ArrayList<String>();
        while (in.hasNext()) {
            String s = in.next();
            if (s.length() > 1)
                theList.add(s);
        } // while
        return theList;
    } // scannerToList()

    public static void main(String[] args) {
        String s = "Here’s a string with several tokens.";
        Scanner theTokens = new Scanner(s);
        List<String> theList = scannerToList(theTokens);
        for (String token: theList)
            System.out.print(token + " ");
        System.out.println();
    } // main()

} // class T2Last
```