Question 1. [5 points] Consider the following static method:

```
public static int mystery(int[] arr) {
  int x = 0;
  for (int i = 0; i < arr.length; i++) {
    x += arr[i];
  }
  for (int i = 0; i < arr.length; i++) {
    x += arr[i];
  }
  return x;
}
```

Let N be the number of elements in the array `arr`. State a worst-case big-O upper bound on the running time of this method in terms of N. Explain your answer briefly.

Question 2. [5 points] Consider the following static method:

```
public static int numNegativeRows(int[][] matrix) {
  int count = 0;
  for (int i = 0; i < matrix.length; i++) {
    int rowSum = 0;
    for (int j = 0; j < matrix[i].length; j++) {
      rowSum += matrix[i][j];
    }
    if (rowSum < 0) { count++; }
  }
  return count;
}
```

Assume that the array `matrix` is square (has the same number of rows and columns), and that N is the number of rows/columns. State a worst-case big-O upper bound on the running time of this method in terms of N. Explain your answer briefly.

Question 3. [5 points] Consider the following static method:

```
public static boolean isUpperTriangular(double[][] matrix) {
  for (int i = 0; i < matrix.length; i++) {
    for (int j = 0; j < i; j++) {
      if (matrix[i][j] != 0.0) { return false; }
    }
  }
  return true;
}
```

Assume that the array `matrix` is square (has the same number of rows and columns), and that N is the number of rows/columns. State a worst-case big-O upper bound on the running time of this method in terms of N. Explain your answer briefly.

Question 4. [5 points] What output is printed by the following code (which begins on the left and continues on the right)?

```
public class Q4 {
  public static void main(
      String[] args) {
    List<Integer> list =
      Arrays.asList(1, 2, 3, 4, 5);

    Iterator<Integer> i =
      list.iterator();
```

```
    System.out.println(i.next());
    System.out.println(i.next());
    if (i.hasNext()) {
      System.out.println(i.next());
    }
    if (i.hasNext()) {
      System.out.println(i.next());
    }
  }
}
```

Question 5. [5 points] Consider the following code:

```
List<Integer> list = Arrays.asList(1, 2, 3, 4, 5);

Iterator<Integer> i = list.iterator();
while (i.hasNext()) {
  System.out.println(i.next());
  System.out.println(i.next());
}
System.out.println("done");
```

What output is printed when this code is executed? Explain your answer briefly.

Question 6. [5 points] Consider the following method:

```
public static void f(String a, String b) {
  int val = 0;
  if (a == b) {
    val = 1;
  }
  System.out.printf("[%s][%s][%d]\n", a, b, val);
}
```

(a) When this method is called, is the output `[X][X][1]` possible? Briefly explain why or why not.

(b) When this method is called, is the output `[X][X][0]` possible? Briefly explain why or why not.

Question 7. [5 points] Consider the following code—the `Box` class on the left, and a code fragment using it on the right:

```
public class Box
    implements Comparable<Box> {
  public int x;

  public Box(int val) { x = val; }

  public int compareTo(Box m) {
    return m.x - this.x;
  }
}
```

```
Box[] boxes = {
  new Box(3),
  new Box(1),
  new Box(2)
};

Arrays.sort(boxes);
for (Box b : boxes) {
  System.out.println(b.x);
}
```

What output is printed by the code on the right?

Question 8. [5 points] Consider the following code—the `Frob` class on the left, and a code fragment using it on the right:

```
public class Frob
    implements Comparator<String> {

  public int compare(
      String lhs, String rhs) {
    return lhs.length() -
           rhs.length();
  }

}
```

```
String[] names =
  { "Alice", "Bob", "Carl" };

Frob f = new Frob();
System.out.println(
  f.compare("Alice", "Bob"));

Arrays.sort(names);
for (String n : names) {
  System.out.println(n);
}
```

What output is printed by the code on the right?

Question 9. [10 points] Complete the following generic method. It should return the minimum of the two parameter values. (Hint: note that the type parameter `E` is guaranteed to implement `Comparable`.)

```
public static<E extends Comparable<E>> E min(E val1, E val2)

                                                    {
```

Programming Questions

To get started, use a web browser to download the zipfile as specified by your instructor. Import it as an Eclipse project using File → Import... → General → Existing Projects into Workspace → Archive file.

Important: You may use the following resources:

- The textbook
- The lecture notes posted on the course web page
- Your previous labs and assignments

Do not open any other files, web pages, etc.

Question 11. [25 points] Complete the implementation of the `Pokemon` class. The class should have three fields:

1. A primary type, whose data type is `PokemonType`
2. A secondary type, whose data type is `PokemonType`
3. A name, whose data type is `String`

There are two constructors:

- The first takes primary type and name as parameters, storing the parameter values in the corresponding fields, and sets the object's secondary type to `PokemonType.NONE`
- The second takes primary type, secondary type, and name, storing the parameter values in the corresponding fields

There should be getters `getPrimaryType`, `getSecondaryType`, and `getName` which return the value of the appropriate field.

The class should implement the `Comparable<Pokemon>` interface. The `compareTo` method should compare first by primary type, then by secondary type (if the primary types are equal), and finally by name (if the primary *and* secondary types are equal). Note that because `PokemonType` is an enumeration, it implicitly implements `Comparable`, so you can compare `PokemonType` values using the `compareTo` method (as you did with `Suit` and `Rank` values in Labs 9 and 10.)

There are unit tests in the `Q11Test` class: make sure they pass.

Question 12. [25 points] In the `Q12` class, implement the `isSorted` static method. The method takes a `List` of elements of type `E`, and returns true if the list is sorted. A list is sorted if and only if for each element other than the first, the previous element is not greater. Note that `E` is guaranteed to implement `Comparable<E>`, so you can call `compareTo` on values of type `E`.

Note: To receive full credit for your solution, your method must complete in $O(N)$ time, where N

is the number of elements in the list. However, you can expect to receive 80% of full credit if your algorithm is correct, but completes in $O(N^2)$ time. (A slow but correct algorithm will get a much better score than an incorrect algorithm.)

There are JUnit tests in the `Q12Test class`. Make sure they pass. Also be aware that any list that is empty or has exactly one element should be considered to be sorted.