1. What does it mean to overload a method? Why is it useful?

To give the same name to two or more methods in the same class or in classes related by inheritance. Overloaded methods must either have different numbers of parameters or have some parameter position that is of differing types in the method headings (i.e., the methods must have different signatures). It is useful because it enables related methods to be called by using a common name. For example, you can have a method that returns the absolute value of an `int`, and another one that returns the absolute value of a `double`. You can name both of these methods based on their function: `abs` (in fact, there are several `abs` methods in Java’s `Math` class). Based on the type of the argument you pass to the method, the compiler can decide which method to use.

2. What does it mean to override a method? Why is it useful?

In a derived class, if you include a method definition that has the same name, the same number, types, and order of parameters, and the same return type as a method already in the base class, this new method definition overrides the method defined in the base class. This new definition replaces the old definition of the method when objects of the derived class receive a call to the method. This enables you to put common functionality in base classes, and more specialized functionality in derived classes, but still use the same method name and parameters to perform the specialized functionality.

3. What is wrong with this code?

```java
int[][] table = new int[15][7];
for (int row = 0; row <= table.length; row++)
    for (int col = 0; col <= table[row].length; col++)
        table[row][col] = 5;
```

The two loop conditions are using `<=` as the comparer. This will result in an array out of bounds exception. They should use `<` instead.
4. Assume you have the following class, with only one method and one constructor defined:

```java
public class Student {
    private String name;

    public Student(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }
}
```

Would the following code compile (in particular, the line that calls the equals method)? Why or why not?

```java
Student std = new Student("Apu");
Student std2 = new Student("James");
if (std.equals(std2))
    System.out.println("They are equal");
```

This code will compile. All classes have `Object` as the base class, and `Object` has an `equals(Object)` method. However, `Object`'s `equals` will probably not behave as desired; you would have to override `equals` if you wanted it to mean that two `Students` are equal if their names are equal.
5. Given the following classes, draw a diagram showing which classes would inherit methods from other classes.

Dog    Quadruped    Creature    Elephant    Biped    Human    Kangaroo

6. Given your diagram in Problem 5, are the following lines of code legal or illegal?

a. legal Quadruped qpd = new Quadruped();
b. illegal Quadruped qpd2 = new Biped();
c. legal Biped bp = new Human();
d. illegal Elephant ele = new Creature();
e. legal Creature cr = new Kangaroo();
f. legal Dog dog = new Dog();
   Creature cr2 = dog;

7. Given your diagram in Problem 5, which class would be the base-most class with these methods?

a. Kangaroo hop()
b. Quadruped walk() // with 4 legs
c. Biped walk() // with 2 legs
d. Elephant moveTrunk()
e. Human checkEmail()
8. Consider the following method:

```java
public static void changeArray(int[][] arr)
{
    for (int row = 0; row < arr.length; row++)
        for (int col = row; col < arr[row].length; col++)
            arr[row][col] = row + col;
}
```

What is contained in the 2D array myArray after the following code executes?

```java
int[][] myArray = new int[5][5];
changeArray(myArray);
```

```
0 1 2 3 4
0 2 3 4 5
0 0 4 5 6
0 0 0 6 7
0 0 0 0 8
```

9. Write a recursive method that finds the largest value in an arbitrary array of integers and returns it.

```java
public static int findLargest(int[] array)
{
    return findLargest(array, 0, array.length - 1);
}

private static int findLargest(int[] array, int start, int end)
{
    if (end - start <= 1)
    {
        return array[start];
    }
    else
    {
        int midIndex = (start + end) / 2;
        int leftLargest = findLargest(array, start, midIndex - 1);
        int rightLargest = findLargest(array, midIndex, end);

        if (leftLargest > rightLargest)
        {
            return leftLargest;
        }
        else
        {
            return rightLargest;
        }
    }
}
10. Write a method that loads an array from a specified text file, sorts it, and returns it. The format of
the text file is that it starts with an integer representing the number of elements followed by that
number of floating point values; all values are whitespace delimited. Suppose that you have access
to a method static void sort(double[] array) in the MergeSort class to perform the
sorting operation.

```java
public static double[] loadAndSort(File inFile)
{
    Scanner sc = null;
    try
    {
        sc = new Scanner(inFile);
    }
    catch (Exception ex)
    {
        System.out.println("Error");
        System.exit(1);
    }

    double[] array = new double[sc.nextInt()];
    for (int i = 0; i < array.length; i++)
    {
        array[i] = sc.nextDouble();
    }

    MergeSort.sort(array);
    return array;
}
```

How many recursive calls deep will the calls to MergeSort.sort(...) go the number of elements in
the array is 256?

8 recursive calls, 9 total calls

8 = \log_2(256)