1. Consider the following class `Thing`, which, unfortunately, doesn’t properly override `equals`, but overloads it instead.

```java
class Thing {
    private int contents;
    public Thing(int value) { this.contents = value; }
    public boolean equals(Thing other) {
        return this.contents == other.contents;
    }
}
```

Now, here is a program that uses class `Thing`. After each `System.out.println` statement, indicate which `equals` method is called (Object.equals or Thing.equals), and whether the statement prints `true` or `false`. Circle the right choices.

```java
public static void main(String[] args) {
    Thing t = new Thing(17);
    Object o = t;
    Thing u = new Thing(17);
    System.out.println(t.equals(o));  // method called: Object.equals   Thing.equals
    // output:  true  false
    System.out.println(t.equals(u));  // method called: Object.equals   Thing.equals
    // output:  true  false
    System.out.println(o.equals(u));  // method called: Object.equals   Thing.equals
    // output:  true  false
    System.out.println(u.equals(o));  // method called: Object.equals   Thing.equals
    // output:  true  false
    System.out.println(u.equals((Thing) o));  // method called: Object.equals   Thing.equals
    // output:  true  false
}
```

(continued on next page)
2. Here is a small class with a correctly overridden `equals` method.

```java
class Blob {
    private int size;  
    private int weight;  
    private String color;

    /** equality for Blobs. Two Blobs are the same
     * if they have the same size and weight. */
    @Override
    public boolean equals(Object o) {
        if (! (o instanceof Blob)) {
            return false;
        }
        Blob b = (Blob) o;
        return this.size == b.size && this.weight == b.weight;
    }
}
```

Below are five possible `hashCode` functions for Blob. For each one indicate if it is incorrect (does not satisfy the specification for `hashCode`), or if it is correct but very poor, or correct and adequate to good. Circle the right answers.

Hints: `^` is the exclusive-or arithmetic operation. Recall that if `a.equals(b)` is true, then `a.hashCode()` must equal `b.hashCode()`.

```java
public int hashCode() { return size; }
```
- Not correct

```java
public int hashCode() { return color.hashCode(); }
```
- Not correct
- Correct but very poor
- Correct and adequate to good

```java
public int hashCode() { return 42; }
```
- Not correct
- Correct but very poor
- Correct and adequate to good

```java
public int hashCode() {
    return size ^ weight ^ color.hashCode();
}
```
- Not correct
- Correct but very poor
- Correct and adequate to good

```java
public int hashCode() { return size ^ weight; }
```
- Not correct
- Correct but very poor
- Correct and adequate to good