Answers and Explanations

Bullets mark each step in the process of arriving at the correct solution.

1. The answer is C.

- Since 9 > 5, we execute the statements after the if and skip the statements after the else.
- 9 + 5 = 14. 14 % 5 = 4 (since 14 / 5 = 2 remainder 4), so we print 4.

2. The answer is B.

- This is a confusing one. We count on indenting to be correct and we use it to evaluate code, but in this case it is deceiving us. Remember that indenting is for humans; Java doesn't care how we indent our code.
- An else clause always binds to the *closest* available if. That means the else belongs to if (value > 15) not to if (value > 10). The code should be formatted like this:

```
if (value > 10)
   if (value > 15)
      value = 0;
   else
      value = 1;
System.out.println("value = " + value);
```

- This makes it clearer that `value = 1` will be printed if the first condition is true (so value has to be greater than 10) and the second condition is false (so value has to be less than or equal to 15). The only value in the answers that fits those criteria is 12, so initValue =12 is the correct answer.

3. The answer is C.

- We start out by using De Morgan's Law (notice that || becomes && when we distribute the !).

  `!(a > b || b <= c)` → `!(a > b) && !(b <= c)`

- !> is the same as <= (don't forget the =), and !<= is the same as >, so

  `!(a > b) && !(b <= c)` → `a <= b && b > c`

4. The answer is D.

- Changing `!(80 < temp)` to its equivalent expression gives `(80 >= temp)`, which is equivalent to `(temp <= 80)`.
- Substituting that back into the original conditional statement gives `if ((temp > 80) && (temp <=80))`, which will always be false.

5. The answer is E.

- The nested `if` segment II and segment III will correctly assign the level based on the student average.
- Segment I will correctly assign the "honors" after the first `if` statement, but as soon as the second `if` statement is evaluated, the condition will be `false` and fall to the `else` statement which will incorrectly assign "regular" to students whose average is 92 and above.

6. The answer is C.

- Rewrite the expressions, filling in true or false for x, y, and z. Remember order of operations. && has precedence over ||.
- true && false && true
  - true && false = false, and since false && anything is false, this one is false.
- true && false || true && false
  - Order of operations is && before || so this simplifies to false || false = false.
- !(true && false) || true
  - We don't need to think about the first half because as soon as we see || true, we know this one is true.

We've found our answer, but let's keep going so we can explain them all.

- !(true || false) && true
  - true || false is true, !true is false. false && anything is false, so this one is false.
- true && false || !true
  - true and false is false, !true is false, so this is false || false, which is false.

7. The answer is E.

- The code shows an example of a dangling else. The indentation implies that the `System.out.println("%%")` is reached when a negative amount is encountered, but that is not the case.
- The `else System.out.println("%%")` statement belongs with the closest unmatched if statement, which is `if (amount <=10)`

8. The answer is C.

- Relational operators are not defined for `Strings`, so the syntax in segment I is not correct.
- The `compareTo` method will return an integer. The expression in the conditional must evaluate to a `boolean`, so the syntax in segment II is not correct.
- If the `compareTo` method returns a negative value, that means `name1` comes alphabetically before `name2`, so segment III is correct.