SELF TEST

1. Given:

```
class Hexy {
  public static void main(String[] args) {
    Integer i = 42;
    String s = (i<40)?"life":(i>50)?"universe":"everything";
    System.out.println(s);
  }
}
```

What is the result?

A. `null`
B. `life`
C. `universe`
D. `everything`
E. Compilation fails
F. An exception is thrown at runtime

2. Given:

```
1. class Comp2 {
2.   public static void main(String[] args) {
3.     float f1 = 2.3f;
4.     float[][] f2 = {{42.0f}, {1.7f, 2.3f}, {2.6f, 2.7f}};
5.     float[] f3 = {2.7f};
6.     Long x = 42L;
7.     // insert code here
8.       System.out.println("true");
9.   }
10. }
```

And the following five code fragments:

```
F1.  if(f1 == f2)
F2.  if(f1 == f2[2][1])
F3.  if(x == f2[0][0])
F4.  if(f1 == f2[1,1])
F5.  if(f3 == f2[2])
```

What is true?

A. One of them will compile, only one will be `true`
B. Two of them will compile, only one will be `true`
C. Two of them will compile, two will be `true`
D. Three of them will compile, only one will be `true`
E. Three of them will compile, exactly two will be `true`
F. Three of them will compile, exactly three will be `true`

3. Given:

```
class Fork {
  public static void main(String[] args) {
    if(args.length == 1 | args[1].equals("test")) {
      System.out.println("test case");
    } else {
      System.out.println("production " + args[0]);
    }
  }
}
```

And the command-line invocation:

```
java Fork live2
```

What is the result?

A. `test case`
B. `production live2`
C. `test case live2`
D. Compilation fails
E. An exception is thrown at runtime

4. Given:

```
class Feline {
  public static void main(String[] args) {
    Long x = 42L;
    Long y = 44L;
    System.out.print(" " + 7 + 2 + " ");
    System.out.print(foo() + x + 5 + " ");
    System.out.println(x + y + foo());
  }
  static String foo() { return "foo"; }
}
```

What is the result?

A. `9 foo47 86foo`

B. `9 foo47 4244foo`

C. `9 foo425 86foo`

D. `9 foo425 4244foo`

E. `72 foo47 86foo`

F. `72 foo47 4244foo`

G. `72 foo425 86foo`

H. `72 foo425 4244foo`

I. Compilation fails

5. Place the fragments into the code to produce the output 33. Note, you must use each fragment exactly once.

CODE:

```
class Incr {
  public static void main(String[] args) {
    Integer x = 7;
    int y = 2;

    x    ___  ___;
    ___  ___  ___;
    ___  ___  ___;
    ___  ___  ___;

    System.out.println(x);
  }
}
```

FRAGMENTS:

y	y	y	y
y	x	x	
-=	*=	*=	*=

6. Given:

```
3. public class Twisty {
4.   { index = 1; }
5.   int index;
6.   public static void main(String[] args) {
7.     new Twisty().go();
8.   }
9.   void go() {
10.    int [][] dd = {{9,8,7}, {6,5,4}, {3,2,1,0}};
11.    System.out.println(dd[index++][index++]);
12.  }
13. }
```

What is the result? (Choose all that apply.)

A. 1

B. 2

C. 4

D. 6

E. 8

F. Compilation fails

G. An exception is thrown at runtime

7. Given:

```
3. public class McGee {
4.   public static void main(String[] args) {
5.     Days d1 = Days.TH;
6.     Days d2 = Days.M;
7.     for(Days d: Days.values()) {
8.       if(d.equals(Days.F)) break;
9.       d2 = d;
10.    }
11.    System.out.println((d1 == d2)?"same old" : "newly new");
12.  }
13.  enum Days {M, T, W, TH, F, SA, SU};
14. }
```

What is the result?

A. `same old`

B. `newly new`

C. Compilation fails due to multiple errors
D. Compilation fails due only to an error on line 7
E. Compilation fails due only to an error on line 8
F. Compilation fails due only to an error on line 11
G. Compilation fails due only to an error on line 13

8. Given:

```
 4. public class SpecialOps {
 5.   public static void main(String[] args) {
 6.     String s = "";
 7.     Boolean b1 = true;
 8.     boolean b2 = false;
 9.     if((b2 = false) | (21%5) > 2) s += "x";
10.     if(b1 || (b2 == true))        s += "y";
11.     if(b2 == true)                s += "z";
12.     System.out.println(s);
13.   }
14. }
```

Which are true? (Choose all that apply.)

A. Compilation fails
B. x will be included in the output
C. y will be included in the output
D. z will be included in the output
E. An exception is thrown at runtime

9. Given:

```
 3. public class Spock {
 4.   public static void main(String[] args) {
 5.     int mask = 0;
 6.     int count = 0;
 7.     if( ((5<7) || (++count < 10)) | mask++ < 10 )   mask = mask + 1;
 8.     if(  (6 > 8) ^ false)                           mask = mask + 10;
 9.     if( !(mask > 1) && ++count > 1)                 mask = mask + 100;
10.     System.out.println(mask + " " + count);
11.   }
12. }
```

Which two are true about the value of mask and the value of count at line 10? (Choose two.)

A. `mask` is 0

B. `mask` is 1

C. `mask` is 2

D. `mask` is 10

E. `mask` is greater than 10

F. `count` is 0

G. `count` is greater than 0

10. Given:

```
 3. interface Vessel { }
 4. interface Toy { }
 5. class Boat implements Vessel { }
 6. class Speedboat extends Boat implements Toy { }
 7. public class Tree {
 8.   public static void main(String[] args) {
 9.     String s = "0";
10.     Boat b = new Boat();
11.     Boat b2 = new Speedboat();
12.     Speedboat s2 = new Speedboat();
13.     if((b instanceof Vessel) && (b2 instanceof Toy))  s += "1";
14.     if((s2 instanceof Vessel) && (s2 instanceof Toy)) s += "2";
15.     System.out.println(s);
16.   }
17. }
```

What is the result?

A. `0`

B. `01`

C. `02`

D. `012`

E. Compilation fails

F. An exception is thrown at runtime

SELF TEST ANSWERS

1. Given:

```
class Hexy {
  public static void main(String[] args) {
    Integer i = 42;
    String s = (i<40)?"life":(i>50)?"universe":"everything";
    System.out.println(s);
  }
}
```

What is the result?

A. `null`
B. `life`
C. `universe`
D. `everything`
E. Compilation fails
F. An exception is thrown at runtime

Answer:

☑ **D** is correct. This is a ternary nested in a ternary with a little unboxing thrown in. Both of the ternary expressions are `false`.

☒ **A, B, C, E,** and **F** are incorrect based on the above. (Objective 7.6)

2. Given:

```
1. class Comp2 {
2.   public static void main(String[] args) {
3.     float f1 = 2.3f;
4.     float[][] f2 = {{42.0f}, {1.7f, 2.3f}, {2.6f, 2.7f}};
5.     float[] f3 = {2.7f};
6.     Long x = 42L;
7.     // insert code here
8.       System.out.println("true");
9.   }
10. }
```

And the following five code fragments:

```
F1.  if(f1 == f2)
F2.  if(f1 == f2[2][1])
F3.  if(x == f2[0][0])
F4.  if(f1 == f2[1,1])
F5.  if(f3 == f2[2])
```

What is true?

A. One of them will compile, only one will be `true`
B. Two of them will compile, only one will be `true`
C. Two of them will compile, two will be `true`
D. Three of them will compile, only one will be `true`
E. Three of them will compile, exactly two will be `true`
F. Three of them will compile, exactly three will be `true`

Answer:

☑ **D** is correct. Fragments `F2`, `F3`, and `F5` will compile, and only `F3` is `true`.
☒ **A, B, C, E,** and **F** are incorrect. `F1` is incorrect because you can't compare a primitive to an array. `F4` is incorrect syntax to access an element of a two-dimensional array. (Objective 7.6)

3. Given:

```
class Fork {
  public static void main(String[] args) {
    if(args.length == 1 | args[1].equals("test")) {
      System.out.println("test case");
    } else {
      System.out.println("production " + args[0]);
    }
  }
}
```

And the command-line invocation:

```
java Fork live2
```

What is the result?

A. `test case`
B. `production live2`

C. `test case live2`
D. Compilation fails
E. An exception is thrown at runtime

Answer:

☑ **E** is correct. Because the short circuit `(||)` is not used, both operands are evaluated. Since `args[1]` is past the args array bounds, an `ArrayIndexOutOfBoundsException` is thrown.

☒ **A, B, C,** and **D** are incorrect based on the above. (Objective 7.6)

4. Given:

```
class Feline {
  public static void main(String[] args) {
    Long x = 42L;
    Long y = 44L;
    System.out.print(" " + 7 + 2 + " ");
    System.out.print(foo() + x + 5 + " ");
    System.out.println(x + y + foo());
  }
  static String foo() { return "foo"; }
}
```

What is the result?

A. `9 foo47 86foo`
B. `9 foo47 4244foo`
C. `9 foo425 86foo`
D. `9 foo425 4244foo`
E. `72 foo47 86foo`
F. `72 foo47 4244foo`
G. `72 foo425 86foo`
H. `72 foo425 4244foo`
I. Compilation fails

Answer:

☑ **G** is correct. Concatenation runs from left to right, and if either operand is a `String`, the operands are concatenated. If both operands are numbers they are added together. Unboxing works in conjunction with concatenation.

☒ **A, B, C, D, E, F, H,** and **I** are incorrect based on the above. (Objective 7.6)

5. Place the fragments into the code to produce the output 33. Note, you must use each fragment exactly once.

CODE:

```
class Incr {
  public static void main(String[] args) {
    Integer x = 7;
    int y = 2;

    x    ___  ___;
    ___  ___  ___;
    ___  ___  ___;
    ___  ___  ___;

    System.out.println(x);
  }
}
```

FRAGMENTS:

y	y	y	y
y	x	x	
-=	*=	*=	*=

Answer:

```
class Incr {
  public static void main(String[] args) {
    Integer x = 7;
    int y = 2;

    x *= x;
    y *= y;
    y *= y;
    x -= y;

    System.out.println(x);
  }
}
```

Yeah, we know it's kind of puzzle-y, but you might encounter something like it on the real exam. (Objective 7.6)

6. Given:

```
3. public class Twisty {
4.   { index = 1; }
5.   int index;
6.   public static void main(String[] args) {
7.     new Twisty().go();
8.   }
9.   void go() {
10.    int [][] dd = {{9,8,7}, {6,5,4}, {3,2,1,0}};
11.    System.out.println(dd[index++][index++]);
12.  }
13. }
```

What is the result? (Choose all that apply.)

A. 1

B. 2

C. 4

D. 6

E. 8

F. Compilation fails

G. An exception is thrown at runtime

Answer:

☑ **C** is correct. Multidimensional arrays' dimensions can be inconsistent, the code uses an initialization block, and the increment operators are both post-increment operators.

☒ **A, B, D, E, F,** and **G** are incorrect based on the above. (Objective 1.3)

7. Given:

```
3. public class McGee {
4.   public static void main(String[] args) {
5.     Days d1 = Days.TH;
6.     Days d2 = Days.M;
7.     for(Days d: Days.values()) {
8.       if(d.equals(Days.F)) break;
9.       d2 = d;
10.    }
11.    System.out.println((d1 == d2)?"same old" : "newly new");
```

```
12.   }
13.   enum Days {M, T, W, TH, F, SA, SU};
14. }
```

What is the result?

A. `same old`

B. `newly new`

C. Compilation fails due to multiple errors

D. Compilation fails due only to an error on line 7

E. Compilation fails due only to an error on line 8

F. Compilation fails due only to an error on line 11

G. Compilation fails due only to an error on line 13

Answer:

☑ **A** is correct. All of this syntax is correct. The for-each iterates through the enum using the `values()` method to return an array. Enums can be compared using either `equals()` or ==. Enums can be used in a ternary operator's Boolean test.

☒ **B, C, D, E, F,** and **G** are incorrect based on the above. (Objective 7.6)

8. Given:

```
 4. public class SpecialOps {
 5.   public static void main(String[] args) {
 6.     String s = "";
 7.     Boolean b1 = true;
 8.     Boolean b2 = false;
 9.     if((b2 = false) | (21%5) > 2) s += "x";
10.     if(b1 || (b2 = true))         s += "y";
11.     if(b2 == true)                s += "z";
12.     System.out.println(s);
13.   }
14. }
```

Which are true? (Choose all that apply.)

A. Compilation fails

B. x will be included in the output

C. y will be included in the output

D. z will be included in the output

E. An exception is thrown at runtime

Answer:

☑ **C** is correct. First of all, boxing takes care of the Boolean. Line 9 uses the modulus operator, which returns the remainder of the division, which in this case is 1. Also, line 9 sets `b2` to false, and it doesn't test `b2`'s value. Line 10 sets `b2` to true, and it doesn't test its value; however, the short circuit operator keeps the expression `b2 = true` from being executed.

☒ **A, B, D,** and **E** are incorrect based on the above. (Objective 7.6)

9. Given:

```
 3. public class Spock {
 4.   public static void main(String[] args) {
 5.     int mask = 0;
 6.     int count = 0;
 7.     if( ((5<7) || (++count < 10)) | mask++ < 10 )    mask = mask + 1;
 8.     if(  (6 > 8) ^ false)                            mask = mask + 10;
 9.     if( !(mask > 1) && ++count > 1)                  mask = mask + 100;
10.     System.out.println(mask + " " + count);
11.   }
12. }
```

Which two answers are true about the value of mask and the value of count at line 10? (Choose two.)

A. `mask` is 0

B. `mask` is 1

C. `mask` is 2

D. `mask` is 10

E. `mask` is greater than 10

F. `count` is 0

G. `count` is greater than 0

Answer:

☑ **C** and **F** are correct. At line 7 the `||` keeps `count` from being incremented, but the `|` allows `mask` to be incremented. At line 8 the ^ returns true only if exactly one operand is true. At line 9 `mask` is 2 and the `&&` keeps `count` from being incremented.

☒ **A, B, D, E,** and **G** are incorrect based on the above. (Objective 7.6)

10. Given:

```
 3. interface Vessel { }
 4. interface Toy { }
 5. class Boat implements Vessel { }
 6. class Speedboat extends Boat implements Toy { }
 7. public class Tree {
 8.   public static void main(String[] args) {
 9.     String s = "0";
10.     Boat b = new Boat();
11.     Boat b2 = new Speedboat();
12.     Speedboat s2 = new Speedboat();
13.     if((b instanceof Vessel) && (b2 instanceof Toy))  s += "1";
14.     if((s2 instanceof Vessel) && (s2 instanceof Toy)) s += "2";
15.     System.out.println(s);
16.   }
17. }
```

What is the result?

A. `0`

B. `01`

C. `02`

D. `012`

E. Compilation fails

F. An exception is thrown at runtime

Answer:

☑ **D** is correct. First, remember that `instanceof` can look up through multiple levels of an inheritance tree. Also remember that `instanceof` is commonly used before attempting a downcast, so in this case, after line 15, it would be possible to say `Speedboat s3 = (Speedboat)b2;`.

☒ **A, B, C, E,** and **F** are incorrect based on the above. (Objective 7.6)