1. Consider the following method:

```
public void reduce(int[] arr, int len)
{
  for (int i = 0; i < len; i++)
  {
    arr[i]--;
  }
  len--;
}
```

What is the output of the following code segment?

```
int[] counts = {3, 2, 1, 0, 0, 0};
int len = 3;
reduce(counts, len);
for (int k : counts)
{
  System.out.print(k + " ");
}
System.out.println(len);
```

(A) `0 1 2 2`
(B) `0 1 2 3 4 5 3`
(C) `2 1 0 0 0 0 2`
(D) `2 1 0 0 0 0 3`
(E) `3 2 1 0 0 0 3`

2. Which of the following statements will result in a syntax error?

(A) `String x = "123";`
(B) `Integer x = "123";`
(C) `Object x = "123";`
(D) `Comparable x = "123";`
(E) All of the above will compile with no errors.

3. What is the result when the following code segment is compiled and executed?

```
int m = 4, n = 5;
double d = Math.sqrt((m + n)/2);
System.out.println(d);
```

(A) Syntax error "sqrt(double) in java.lang.Math cannot be applied to int"
(B) `1.5` is displayed
(C) `2.0` is displayed
(D) `2.1213203435596424` is displayed
(E) `ClassCastException`

4. Suppose we have found a compiled Java class, `Fun.class`, but we do not have its source code. We have discovered that a statement

```
Fun fun = new Fun(100);
```

compiles with no errors. Which of the following statements, if it compiles correctly, will convince us that `Fun` implements `Comparable<Fun>`?

(A) `Comparable<Fun> c = fun;`
(B) `System.out.print(fun.compareTo(0));`
(C) `System.out.print(fun.compareTo(fun));`
(D) `System.out.print(fun.compareTo(new Fun(99)));`
(E) None of the above

5. What is the value of `product` after the following code segment is executed?

```
int[] factors = {2, 3, 4, 7, 2, 5};
int product = 1;
for (int i = 1; i < factors.length; i += 2)
{
  product *= (factors[i] % factors[i - 1]);
}
```

(A) 0
(B) 1
(C) 2
(D) 3
(E) 5

6. Given that `x` is *true*, `y` is *true*, and `z` is *false*, which of the following expressions will evaluate to *false*?

(A) `(x && y) || z`
(B) `(x || y) && z`
(C) `y || (x && z)`
(D) `x || (y && z)`
(E) `x && (y || z)`

7. Consider the following method:

```
public String encrypt(String word)
{
  int pos = word.length() / 2;
  if (pos >= 1)
  {
    word = encrypt(word.substring(pos)) +
           encrypt(word.substring(0, pos));
  }
  return word;
}
```

What is the contents of the string returned by `encrypt("SECRET")`?

(A) `TERCES`
(B) `TSECRE`
(C) `RETSEC`
(D) `CESTER`
(E) `ETRECS`

8. What is the output of the following code segment?

```
String url = "http://www.usa.gov";
int pos = url.indexOf("http://");
if (pos >= 0)
{
  System.out.println("<" + url.substring(0, pos) + ">");
}
else
{
  System.out.println("not found");
}
```

(A) `<>`
(B) `<www.usa.gov>`
(C) `<http://www.usa.gov>`
(D) `not found`
(E) `IndexOutOfBoundsException`

9. At the county fair, prizes are awarded to the five heaviest pigs. More than 5,000 pigs are entered, and their records are stored in an array. Which of these would be the most efficient way of finding the records of the five heaviest pigs?

(A) Selection Sort
(B) Selection Sort terminated after the first five iterations
(C) Insertion Sort
(D) Insertion Sort terminated after the first five iterations
(E) Mergesort

10. Suppose the following three classes are defined and each has a "no-args" constructor (a constructor that takes no parameters):

```
public class Vehicle {...}
public class Car extends Vehicle {...}
public class Buick extends Car {...}
```

Which of the following is NOT a legal statement?

(A) `Vehicle vehicle = new Car();`
(B) `Vehicle vehicle = new Buick();`
(C) `Car car = new Buick();`
(D) `Car car = new Vehicle();`
(E) `Buick buick = new Buick();`

11. Consider the following method:

```
// returns the number of times the digit d occurs in the
// decimal representation of n
// precondition:  n and d are non-negative integers
private int findDigit(int n, int d)
{
  int count = 0;
  < statement1 >

  while (n > 0)
  {
    if (n % 10 == d)
    {
      count++;
    }
    < statement2 >
  }

  return count;
}
```

Which of the following could replace < *statement1* > and < *statement2* > to make `findDigit` work as specified?

	< *statement1* >	< *statement2* >
(A)	`if (n == 0) return 1;`	`n /= 10;`
(B)	`if (n == 0) return 1;`	`d *= 10;`
(C)	`if (d == 0) count++;`	`n -= n % 10;`
(D)	`if (n == 0 && d != 0) return 0;`	`n *= 10;`
(E)	`if (n == 0 && d == 0) count++;`	`n /= 10;`

12. Which of the following tasks is made easier when information hiding is practiced?

    I. Implementing IS-A relationships for classes

    II. Making changes to the implementation of one of the classes in a project

    III. Producing specifications for individual programmers working on the same project

(A) I only
(B) II only
(C) I and II
(D) II and III
(E) I, II, and III

13. The two versions of the `search` method shown below are both intended to return `true` if `ArrayList list` contains the target value, `false` otherwise.

Version 1:

```
public boolean search(ArrayList<Object> list, Object target)
{
  for (int k = 0; k < list.size(); k++)
  {
    if (target.equals(list.get(k)))
      return true;
  }
  return false;
}
```

Version 2:

```
public boolean search(ArrayList<Object> list, Object target)
{
  boolean found = false;

  for (int k = 0; k < list.size(); k++)
  {
    if (target.equals(list.get(k)))
      found = true;
  }
  return found;
}
```

Which of the following statements about the two versions of `search` is true?

(A) Only Version 1 works as intended.
(B) Only Version 2 works as intended.
(C) Both versions work as intended; Version 1 is often more efficient than Version 2.
(D) Both versions work as intended; Version 2 is often more efficient than Version 1.
(E) Both versions work as intended; the two versions are always equally efficient.

Questions 14-15 refer to the following implementation of Mergesort:

```
public class Mergesort
{
  // returns a new array which holds the values
  // arr[m], arr[m+1], ... arr[n] arranged in ascending order
  // precondition: 0 <= m <= n
  public static int[] sort(int[] arr, int m, int n)
  {
    int[] result = new int[n - m + 1];

    if (m == n)
    {
      result[0] = arr[m];
    }
    else
    {
      int mid = (n + m) / 2;
      int[] result1 = sort(arr, m, mid);
      int[] result2 = sort(arr, mid + 1, n);
      result = merge(result1, result2);
    }

    return result;
  }

  // merges arr1 and arr2 in ascending order and returns the
  // resulting array
  private static int[] merge(int[] arr1, int[] arr2)
  { /* implementation not shown */ }
}
```

14. If `int[] arr` holds eight values and `Mergesort(arr, 0, 7)` is called, how many times in total will `Mergesort`'s `merge` method will be called?

    (A) 1
    (B) 3
    (C) 7
    (D) 8
    (E) 15

15. If `Mergesort.sort(arr, 0, 999)` takes on average 40 ms and `Mergesort.merge(arr1, arr2)` takes on average `0.01*(arr1.length + arr2.length)`, what is the average run time for `Mergesort.sort(arr, 0, 1999)`?

    (A) 50 ms
    (B) 100 ms
    (C) 160 ms
    (D) 170 ms
    (E) 180 ms

16. Consider the following classes:

```
public class APTestResult
{
  private String subject;
  private int score;

  public int getScore() { return score; }

  // ... constructors and other methods not shown
}

public class APScholar
{
  private String name;
  private int id;
  private ArrayList<APTestResult> exams;

  public ArrayList<APTestResult> getExams() { return exams;

  // ... constructors and other methods not shown
}
```

Given

```
APScholar[] list = new APScholar[100];
```

which of the following expressions correctly represents the third AP score of the sixth AP Scholar in `list`?

(A) `list[5].exams[2].score`
(B) `list[5].exams.getScore(2)`
(C) `list[5].exams[2].getScore()`
(D) `list[5].getExams(2).getScore()`
(E) `list[5].getExams().get(2).getScore()`

17. What is printed when the following code segment is executed?

```
ArrayList<Integer> list = new ArrayList<Integer>();
list.add(new Integer(1));
list.add(new Integer(2));
for (int i = 1; i <= 3; i++)
{
  list.add(i, new Integer(i));
}
System.out.println(list);
```

(A) `[1, 1, 2, 2, 3]`
(B) `[1, 1, 2, 3, 2]`
(C) `[1, 2, 1, 2, 3]`
(D) `[1, 2, 3, 1, 2]`
(E) `IndexOutOfBoundsException`

Questions 18-21 refer to the abstract class `House` and its subclass `HouseForSale`:

```
public abstract class House implements Comparable<House>
{
  private int mySize;

  public House(int size) { mySize = size; }
  public int getSize() { return mySize; }
  public void setSize(int size) { mySize = size; }

  public int compareTo(House other)
  {
    return getSize() - other.getSize();
  }

  public abstract int getPrice();
}

public class HouseForSale extends House
{
  private int myPrice;

  public HouseForSale(int size, int price)
  {
    // ... missing statement
    myPrice = price;
  }

  public int getPrice() { return myPrice; }

  public int compareTo(House other)
  {
    return getPrice() - ((HouseForSale)other).getPrice();
  }

  // ... other constructors, methods, and fields
  //       not shown
}
```

18. Which of the following is the most appropriate replacement for < *missing statement* > in `HouseForSale`'s constructor?

    (A) `mySize = size;`
    (B) `setSize(size);`
    (C) `super.setSize(size);`
    (D) `super(size);`
    (E) `super = new House(size);`

19. Suppose that while coding `HouseForSale` the programmer accidentally misspelled "compareTo" in his class. What will happen when he tries to compile and run his class and the following statements in a client class?

```
HouseForSale house1 = new HouseForSale(2000, 129000);
HouseForSale house2 = new HouseForSale(1800, 149000);
System.out.println(house1.compareTo(house2));
```

(A) A syntax error "undefined compareTo method" in the `HouseForSale` class
(B) A syntax error "HouseForSale should be declared abstract"
(C) The code compiles with no errors and displays `200`.
(D) The code compiles with no errors but generates a `NoSuchMethodException`.
(E) The code compiles with no errors but generates a `ClassCastException`.

20. If the classes `House` and `HouseForSale` compile with no problems, which of the following declarations will result in a syntax error?

(A)
```
House[] houses = new House[2];
```

(B)
```
HouseForSale[] houses = {new House(2000), new House(1800)};
```

(C)
```
House[] houses = {new HouseForSale(2000, 129000),
                  new HouseForSale(1800, 149000)};
```

(D)
```
Comparable[] houses = {new HouseForSale(2000, 129000),
                       new HouseForSale(1800, 149000)};
```

(E)
```
HouseForSale[] houses = {new HouseForSale(2000, 129000),
                         new HouseForSale(1800, 149000)};
```

21. Which of the following is the most appropriate way to define the `getSize` method in `HouseForSale`?

(A) `public int getSize() { return mySize; }`
(B) `public int getSize() { return super.mySize; }`
(C) `public int getSize() { return super(mySize); }`
(D) `public int getSize() { return super.getSize(); }`
(E) No definition is necessary because the same code is already written in `House`.

22. The Binary Search algorithm is designed to work with an array sorted in ascending order. Under which of the following circumstances will the algorithm find a given target value even if the array is not sorted?

I. The array has an odd number of elements and the target value is located exactly in the middle of the array.

II. The array is partially sorted: the left third of the array has values all in ascending order and the target value is among them.

III. The array is partially sorted: all the values to the left of the target are smaller than the target and all the values to the right of the target are larger than the target.

(A) I only
(B) I and II
(C) I and III
(D) II and III
(E) I, II, and III

23. Consider the following method:

```
private double compute(int x, int y)
{
  double r = 0;

  if (!(y == 0 || x / y <= 2))
    r = 1 / ((x - 2*y) * (2*x - y));

  return r;
}
```

For which of the following values of `x` and `y` will `compute(x, y)` throw an exception?

(A) `x = 0, y = 0`
(B) `x = 1, y = 2`
(C) `x = 2, y = 1`
(D) `x = 3, y = 5`
(E) None of the above

Questions 24-29 refer to the code from the GridWorld case study.

24. How does a `Bug` act if there is a `Rock` directly in front of it in the grid?

(A) The `Bug` is removed from the grid.
(B) The `Bug` remains in its current state — no action is taken.
(C) The `Rock` is removed, and the `Bug` moves forward, leaving a new `Flower` in its old location.
(D) The `Bug` turns 180 degrees.
(E) The `Bug` turns 45 degrees to the right.

25. Consider the following subclass of `Bug`:

```
public class SpinningBug extends Bug
{
  public void act()
  {
    super.act();
    turn();
  }

  public void turn()
  {
    setDirection(getDirection() + Location.LEFT);
  }
}
```

If a `SpinningBug` is put into a grid of actors, how does it act?

(A) The `SpinningBug` throws a `NoSuchMethodException`, because the `canMove` and `move` methods are undefined.
(B) The `SpinningBug` acts like a regular `Bug` but turns 90 degrees to the left after each move.
(C) If the `SpinningBug` can move, it moves forward, then turns 90 degrees to the left; otherwise the bug turns 180 degrees.
(D) If the `SpinningBug` can move, it moves forward, then turns 90 degrees to the left; otherwise the bug turns 45 degrees to the left.
(E) If the `SpinningBug` can move (that is, the location in front is valid and empty), it moves forward, then turns 90 degrees to the left; otherwise the bug turns 45 degrees to the right.

26. Which of the following code segments will compile with no errors?

I.
```
Actor a = new Actor();
a.setColor(Color.RED);
```

II.
```
Actor a = new Bug(Color.GREEN);
```

III.
```
Actor a = new Bug();
a.setColor(Color.BLUE);
```

(A) I only
(B) II only
(C) I and II
(D) II and III
(E) I, II, and III

27. When a `ChameleonCritter` object is first created, what are its initial color and direction?

(A) blue color and north
(B) red color and north
(C) blue color and random direction
(D) random color and random direction
(E) The color passed to `ChameleonCritter`'s constructor as a parameter and random direction

28. Which of the following is NOT an example of polymorphism?

(A) The appropriate `act` method is called for `Rocks`, `Bugs`, `Flowers`, and `Critters`
(B) `BoxBug`'s `act` method calls `move` and `turn` inherited from `Bug`
(C) `ChameleonCritter`'s `act` method, inherited from `Critter`, calls `ChameleonCritter`'s `processActors` and `makeMove`
(D) `Bug`'s `canMove` method calls the appropriate `Grid`'s `isValid` for different implementations of `Grid`
(E) All of the above are examples of polymorphism.

29. Consider the following class:

```
public class IntGrid extends BoundedGrid<Integer>
{
  // Constructors not shown

  // Increments Integer objects in all occupied locations by 1
  public void increment()
  {
    < missing code >
  }
}
```

Which of the following code segments could replace < *missing code* > so that the method `increment` works as specified?

(A)

```
for (Location loc : getOccupiedLocations())
{
  Integer i = get(loc);
  i = new Integer(i.intValue() + 1);
}
```

(B)

```
for (Location loc : grid.getOccupiedLocations())
{
  Integer i = grid.get(loc);
  i.setValue(i.intValue() + 1);
}
```

(C)

```
for (Location loc : getOccupiedLocations())
{
  Integer i = get(loc);
  i.setValue(i.intValue() + 1);
  put(loc, i);
}
```

(D)

```
for (Location loc : getOccupiedLocations())
  put(loc, new Integer(get(loc).intValue() + 1));
```

(E)

```
for (Location loc : grid.getOccupiedLocations())
  grid.put(loc, new Integer(grid.get(loc).intValue() + 1));
```

30. In a regular pentagon, the ratio of the length of a diagonal to the length of a side is equal to the Golden Ratio (defined as $\frac{1+\sqrt{5}}{2} \approx 1.618$). Consider the following class `Pentagon`, which represents a regular pentagon:

```
public class Pentagon
{
  public static final double goldenRatio =
                                   (1 + Math.sqrt(5.0)) / 2;
  private double side;

  public Pentagon (double x)
  {
    side = x;
  }

  public double getDiagonalLength()
  {
    return side * goldenRatio;
  }
}
```

Which of the following code segments will compile with no syntax errors and display the correct length of a diagonal in a regular pentagon with side 3.0?

I.
```
System.out.println(3/2 * (1 + Math.sqrt(5.0)));
```

II.
```
System.out.println(3.0 * Pentagon.goldenRatio);
```

III.
```
Pentagon p = new Pentagon(3);
System.out.println(p.getDiagonalLength());
```

(A) I only
(B) II only
(C) III only
(D) I and II
(E) II and III

31. Suppose a programmer has written a method that implements Insertion Sort for an array of integers. The method has no preconditions. Which of the following is NOT a useful test for this method?

(A) An array of length 5 with random values
(B) An array of length 5 with values sorted in ascending order
(C) An empty array (length 0)
(D) An array of length 1
(E) All of the above are useful tests.

32. Consider the following code segment with a missing "for" loop:

```
ArrayList<String> letters = new ArrayList<String>();

letters.add("A");
letters.add("B");
letters.add("C");

< missing "for" loop >

System.out.println(letters);
```

Suppose, when executed, the above code segment displays

```
[A*, B*, C*]
```

Which of the following could replace < *missing "for" loop* >?

I.
```
for (int i = 0; i < letters.size(); i++)
{
  letters.set(i, letters.get(i) + "*");
}
```

II.
```
for (int i = 0; i < letters.size(); i++)
{
  String s = letters.get(i);
  s = s + "*";
}
```

III.
```
for (String s : letters)
{
  s = s + "*";
}
```

(A) I only
(B) II only
(C) I and II
(D) II and III
(E) I, II, and III

33. If class `C` implements interface `I` and has a no-args constructor (a constructor that takes no parameters), which of the following statements results in a syntax error?

(A) `C x = new C();`
(B) `I x = new C();`
(C) `C[] x = new C[10];`
(D) `I[] x = new I[10];`
(E) All of the above compile with no errors.

34. Consider the following method:

```
public void printNumbers(int n)
{
  for (int i = 1; i <= n - 1; i++)
    printNumbers(i);
  System.out.print(n + " ");
}
```

What is the output when `printNumbers(3)` is called?

(A) `1 2`
(B) `1 2 3`
(C) `1 1 2 3`
(D) `1 1 2 1 2 3`
(E) `1 1 2 1 1 2 3`

35. Consider two different designs for a data structure to hold the total number of home runs hit in a season by baseball players. There are *n* players (n > 1000), and each total is in the range from 0 to 80.

Design A: Use an array of length 81. Each index into the array corresponds to a number of home runs, and each element of the array is a reference to a list containing the names of the players who hit that many home runs, in no particular order.

Design B: Use an array of length *n* so that each element of the array corresponds to one player. Each element of the array is an object that represents a player, holding his name and the number of home runs he has hit. The elements of the array are sorted alphabetically by player name.

This data structure will be used to support three operations:

Operation 1: Print the names of all players who hit over 50 home runs.

Operation 2: Given a player's name, look up that player's home run total.

Operation 3: Given the names of two players, determine whether they hit the same number of home runs.

Which of the three operations could be performed more efficiently using Design A rather than Design B?

(A) Operation 1 only
(B) Operation 2 only
(C) Operation 3 only
(D) Operations 1 and 2
(E) Operations 2 and 3

36. Consider the following class:

```
public class NumList
{
  private ArrayList<Integer> list;

  public NumList() { list = new ArrayList<Integer>(); }
  public void add(int value) { /* implementation not shown */ }
  public void remove(int value)
  { /* implementation not shown */ }
  public void shuffle() { /* implementation not shown */ }
  public String toString() { /* implementation not shown */ }
}
```

Suppose we plan to implement the `NumList` class by first coding and testing the constructor and two other methods. Which pair of methods can be conveniently chosen to be developed and tested first?

(A) `add` and `toString`
(B) `toString` and `remove`
(C) `add` and `remove`
(D) `shuffle` and `remove`
(E) `add` and `shuffle`

37. Suppose class `C` has a private `int` data field `value`:

```
public class C
{
  private int value;
  // ... other fields, constructors, and methods not shown
}
```

Suppose we have a method

```
public static int compare(C x, C y)
{ return x.value - y.value; }
```

and we need to find "home" for it: place it into some class. Where can we place this method so that it compiles with no errors?

(A) Only into `C`
(B) Only into `C` or any subclass of `C`
(C) Only into `C` or any superclass of `C`
(D) Into any class
(E) This method will always cause a syntax error, no matter what class we place it in.

38. Consider the following class:

```
public class ArrayProcessor
{
  public static void run(int[] arr)
  {
    for (int i = 0; i < arr.length; i++)
    {
      for (int j = arr.length - 1; j > i; j--)
      {
        if (arr[j] < arr[i])
        {
          swap(arr, i, j);
        }
      }
    }
  }

  private static void swap(int[] arr, int i, int j)
  {
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
  }
}
```

How many times will `ArrayProcessor`'s `swap` method be called when the following code segment is executed?

```
int[] counts = {1, 2, 3, 4, 5, 0};
ArrayProcessor.run(counts);
```

(A) 1
(B) 5
(C) 15
(D) 30
(E) 35

Questions 39-40 refer to a project that includes the following classes:

```
public class OrderItem
{
  private String itemName;
  private int price, quantity;

  public OrderItem (String name, int pr, int qty)
  { itemName = name; price = pr; quantity = qty; }

  public int getExtendedPrice() { return price * quantity; }

  // ... other constructors and methods not shown
}

public class Order
{
  private ArrayList<OrderItem> items;

  public Order() { items = new ArrayList<OrderItem>(); }

  public int getTotal()
  {
    int total = 0;

    for (OrderItem item : items)
    {
      total += item.getExtendedPrice();
    }
    return total;
  }

  public void add(OrderItem item) { items.add(item); }
}
```

The project designer has instructed the programmer to modify the code as follows: to introduce

```
public interface Priced
{
  int getExtendedPrice();
}
```

into the project, add `implements Priced` to the `OrderItem` class header, and replace `OrderItem` with `Priced` everywhere in the `Order` class.

39. Which design principle is applied here, and which Java feature makes it possible for the modified code to work?

    (A) Encapsulation and polymorphism
    (B) Abstraction and encapsulation
    (C) Abstraction and polymorphism
    (D) Information hiding and encapsulation
    (E) Information hiding and Java Virtual Machine

40. Which of the following are good reasons for this change?

I. In some future version of the project, the `items` list in an `Order` object may hold items of the type of a subclass of `OrderItem`.

II. In some future version of the project, different types of `Priced` objects can be intermixed in the `items` list in an `Order` object.

III. The `Order` class can be reused in other projects dealing with a different type of `Priced` items.

(A) I only
(B) II only
(C) I and II
(D) II and III
(E) I, II, and III