1. A large Java program was tested extensively and no errors were found. What can be concluded?
   (A) All of the preconditions in the program are correct.
   (B) All of the postconditions in the program are correct.
   (C) The program may have bugs.
   (D) The program has no bugs.
   (E) Every method in the program may safely be used in other programs.

Questions 2–4 refer to the `Worker` class below:

```
public class Worker
{
    private String myName;
    private double myHourlyWage;
    private boolean isUnionMember;

    //constructors

    public Worker()
    { /* implementation not shown */ }

    public Worker(String name, double hourlyWage, boolean union)
    { /* implementation not shown */ }

    //accessors getName, getHourlyWage, getUnionStatus not shown ...

    //modifiers

    //Permanently increase hourly wage by amt.
    public void incrementWage(double amt)
    { /* implementation of incrementWage */ }

    //Switch value of isUnionMember from true to false and vice versa.
    public void changeUnionStatus()
    { /* implementation of changeUnionStatus */ }
}
```

2. Refer to the `incrementWage` method. Which of the following is a correct `/* implementation of incrementWage */`?
   (A) `return myHourlyWage + amt;`
   (B) `return getHourlyWage() + amt;`
   (C) `myHourlyWage += amt;`
   (D) `getHourlyWage() += amt;`
   (E) `myHourlyWage = amt;`

3. Consider the method `changeUnionStatus`. Which is a correct `/* implementation of changeUnionStatus */`?

I
```
if (isUnionMember)
    isUnionMember = false;
else
    isUnionMember = true;
```

II
```
isUnionMember = !isUnionMember;
```

III
```
if (isUnionMember)
    isUnionMember = !isUnionMember;
```

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

4. A client method `computePay` will return a worker's pay based on the number of hours worked.

```
//Precondition:  Worker w has worked the given number of hours.
//Postcondition: Returns amount of pay for Worker w.
public static double computePay(Worker w, double hours)
{ /* code */ }
```

Which replacement for `/* code */` is correct?

(A) `return myHourlyWage * hours;`
(B) `return getHourlyWage() * hours;`
(C) `return w.getHourlyWage() * hours;`
(D) `return w.myHourlyWage * hours;`
(E) `return w.getHourlyWage() * w.hours;`

5. Consider this program segment. You may assume that `wordList` has been declared as `ArrayList<String>`.

```
for (String s : wordList)
    if (s.length() < 4)
        System.out.println("SHORT WORD");
```

What is the maximum number of times that `SHORT WORD` can be printed?

(A) `3`
(B) `4`
(C) `wordList.size()`
(D) `wordList.size() - 1`
(E) `s.length()`

6. Refer to the following method.

```
public static int mystery(int n)
{
    if (n == 1)
        return 3;
    else
        return 3 * mystery(n - 1);
}
```

What value does `mystery(4)` return?

(A) 3
(B) 9
(C) 12
(D) 27
(E) 81

7. Refer to the following declarations:

```
String[] colors = {"red", "green", "black"};
ArrayList<String> colorList = new ArrayList<String>();
```

Which of the following correctly assigns the elements of the `colors` array to `colorList`? The final ordering of colors in `colorList` should be the same as in the `colors` array.

I
```
for (String col : colors)
    colorList.add(col);
```

II
```
for (String col : colorList)
    colors.add(col);
```

III
```
for (int i = colors.length - 1; i >= 0; i--)
    colorList.add(i, colors[i]);
```

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

Questions 8 and 9 refer to the classes `Address` and `Customer` given below.

```
public class Address
{
    private String myStreet;
    private String myCity;
    private String myState;
    private int myZipCode;

    //constructor
    public Address(String street, String city, String state, int zipCode)
    { /* implementation not shown */ }

    //accessors

    public String getStreet()
    { /* implementation not shown */ }

    public String getCity()
    { /* implementation not shown */ }

    public String getState()
    { /* implementation not shown */ }

    public int getZipCode()
    { /* implementation not shown */ }
}

public class Customer
{
    private String myName;
    private String myPhone;
    private Address myAddress;
    private int myID;

    //constructor
    public Customer(String name, String phone, Address addr, int ID)
    { /* implementation not shown */ }

    //accessors

    //Returns address of this customer.
    public Address getAddress()
    { /* implementation not shown */ }

    public String getName()
    { /* implementation not shown */ }

    public String getPhone()
    { /* implementation not shown */ }

    public int getID()
    { /* implementation not shown */ }
}
```

8. Which of the following correctly creates a `Customer` object `c`?

I
```
Address a = new Address("125 Bismark St", "Pleasantville",
    "NY", 14850);
Customer c = new Customer("Jack Spratt", "747-1674", a, 7008);
```

II
```
Customer c = new Customer("Jack Spratt", "747-1674",
    "125 Bismark St, Pleasantville, NY 14850", 7008);
```

III
```
Customer c = new Customer("Jack Spratt", "747-1674",
    new Address("125 Bismark St", "Pleasantville", "NY", 14850),
    7008);
```

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

9. Consider an `AllCustomers` class that has private instance variable

```
private Customer[] custList;
```

Given the ID number of a particular customer, a method of the class, `locate`, must find the correct `Customer` record and return the name of that customer. Here is the method `locate`:

```
/* Precondition:  custList contains a complete list of Customer
 *                objects. idNum matches the ID number data member
 *                of one of the Customer objects.
 * Postcondition: The name of the customer whose ID number
 *                matches idNum is returned.  */
public String locate(int idNum)
{
    for (Customer c : custList)
        if (c.getID() == idNum)
            return c.getName();
    return null;     //idNum not found
}
```

A more efficient algorithm for finding the matching `Customer` object could be used if

(A) `Customer` objects were in alphabetical order by name.
(B) `Customer` objects were sorted by phone number.
(C) `Customer` objects were sorted by ID number.
(D) the `custList` array had fewer elements.
(E) the `Customer` class did not have an `Address` data member.

10. Often the most efficient computer algorithms use a divide-and-conquer approach, for example, one in which a list is repeatedly split into two pieces until a desired outcome is reached. Which of the following use a divide-and-conquer approach?

I Mergesort
II Insertion sort
III Binary search

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

11. In Java, a variable of type `int` is represented internally as a 32-bit signed integer. Suppose that one bit stores the sign, and the other 31 bits store the magnitude of the number in base 2. In this scheme, what is the largest value that can be stored as type `int`?

(A) 2^{32}
(B) $2^{32} - 1$
(C) 2^{31}
(D) $2^{31} - 1$
(E) 2^{30}

12. Refer to method `removeWord`.

```
//Precondition:  wordList is an ArrayList of String objects.
//Postcondition: All occurrences of word have been removed from
                 wordList.
public static void removeWord(ArrayList<String> wordList,
                                                 String word)
{
    for (int i = 0; i < wordList.size(); i++)
        if ((wordList.get(i)).equals(word))
            wordList.remove(i);
}
```

The method does not always work as intended. Consider the method call

```
removeWord(wordList, "cat");
```

For which of the following lists will this method call fail?

(A) `The cat sat on the mat`
(B) `The cat cat sat on the mat mat`
(C) `The cat sat on the cat`
(D) `cat`
(E) `The cow sat on the mat`

13. What will be output by this code segment?

```
for (int i = 5; i > 0; i--)
{
    for (int j = 1; j <= i; j++)
        System.out.print(j * j + " ");
    System.out.println();
}
```

(A)
```
1
1 4
1 4 9
1 4 9 16
1 4 9 16 25
```

(B)
```
1 4 9 16 25
1 4 9 16
1 4 9
1 4
1
```

(C)
```
25 16 9 4 1
25 16 9 4
25 16 9
25 16
25
```

(D)
```
25
25 16
25 16 9
25 16 9 4
25 16 9 4 1
```

(E)
```
1 4 9 16 25
1 4 9 16 25
1 4 9 16 25
1 4 9 16 25
1 4 9 16 25
```

14. Consider two different ways of storing a set of nonnegative integers in which there are no duplicates.

Method One: Store the integers explicitly in an array in which the number of elements is known. For example, in this method, the set {6, 2, 1, 8, 9, 0} can be represented as follows:

Method Two: Suppose that the range of the integers is 0 to MAX. Use a boolean array indexed from 0 to MAX. The index values represent the possible values in the set. In other words, each possible integer from 0 to MAX is represented by a different position in the array. A value of true in the array means that the corresponding integer is in the set, a value of false means that the integer is not in the set. For example, using this method for the same set above, {6, 2, 1, 8, 9, 0}, the representation would be as follows (T = true, F = false):

The following operations are to be performed on the set of integers:

I Search for a target value in the set.
II Print all the elements of the set.
III Return the number of elements in the set.

Which statement is *true*?

(A) Operation I is more efficient if the set is stored using Method One.
(B) Operation II is more efficient if the set is stored using Method Two.
(C) Operation III is more efficient if the set is stored using Method One.
(D) Operation I is equally efficient for Methods One and Two.
(E) Operation III is equally efficient for Methods One and Two.

15. An algorithm for finding the average of N numbers is

$$\text{average} = \frac{\text{sum}}{N}$$

where N and sum are both integers. In a program using this algorithm, a programmer forgot to include a test that would check for N equal to zero. If N is zero, when will the error be detected?

(A) At compile time
(B) At edit time
(C) As soon as the value of N is entered
(D) During run time
(E) When an incorrect result is output

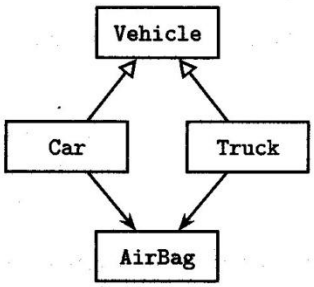
16. What is wrong with this interface?

```
public interface Bad
{
    void someMethod(String password)
    {
        System.out.println("Psst! The password is " + password);
    }
}
```

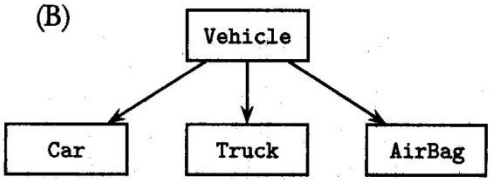
(A) A method in an interface should be declared `public`.
(B) A method in an interface should be declared `abstract`.
(C) There should not be a method implementation.
(D) There should be a class implementation provided.
(E) There should not be any method parameters.

17. Consider a program that deals with various components of different vehicles. Which of the following is a reasonable representation of the relationships among some classes that may comprise the program? Note that an open up-arrow denotes an inheritance relationship and a down-arrow denotes a composition relationship.

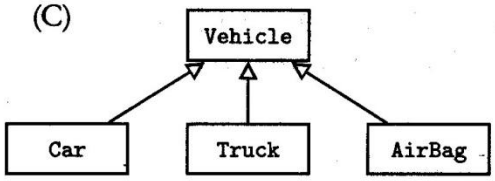
(A)



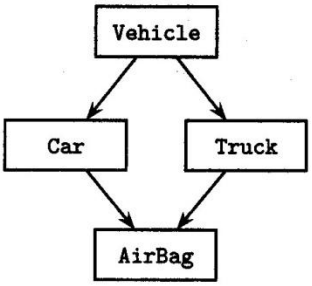
(B)



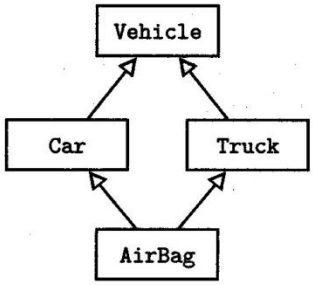
(C)



(D)



(E)



18. Consider the following program segment:

```
//Precondition: a[0]...a[n-1] is an initialized array of
//              integers, 0 < n <= a.length.
    int c = 0;
    for (int i = 0; i < n; i++)
        if (a[i] >= 0)
        {
            a[c] = a[i];
            c++;
        }
    n = c;
```

Which is the best postcondition for the segment?

(A) `a[0]...a[n-1]` has been stripped of all positive integers.
(B) `a[0]...a[n-1]` has been stripped of all negative integers.
(C) `a[0]...a[n-1]` has been stripped of all nonnegative integers.
(D) `a[0]...a[n-1]` has been stripped of all occurrences of zero.
(E) The updated value of `n` is less than or equal to the value of `n` before execution of the segment.

19. If `a`, `b`, and `c` are integers, which of the following conditions is sufficient to *guarantee* that the expression

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

evaluates to true?

(A) `a < c`
(B) `a < b`
(C) `a > b`
(D) `a == b`
(E) `a == c`

20. Airmail Express charges for shipping small packages by integer values of weight. The charges for a weight w in pounds are as follows:

$0 < w \leq 2$	\$4.00
$2 < w \leq 5$	\$8.00
$5 < w \leq 20$	\$15.00

The company does not accept packages that weigh more than 20 pounds. Which of the following represents the best set of data (weights) to test a program that calculates shipping charges?

(A) 0, 2, 5, 20
(B) 1, 4, 16
(C) −1, 1, 2, 3, 5, 16, 20
(D) −1, 0, 1, 2, 3, 5, 16, 20, 22
(E) All integers from −1 through 22

Questions 21–22 are based on the following class declaration:

```
public class AutoPart
{
    private String myDescription;
    private int myPartNum;
    private double myPrice;

    //constructor
    public AutoPart(String description, int partNum, double price)
    { /* implementation not shown */ }

    //accessors

    public String getDescription()
    { return myDescription; }

    public int getPartNum()
    { return myPartNum; }

    public double getPrice()
    { return myPrice; }
}
```

21. This question refers to the `findCheapest` method below, which occurs in a class that has an array of `AutoPart` as one if its private data fields:

```
private AutoPart[] allParts;
```

The `findCheapest` method examines an array of `AutoPart` and returns the part number of the `AutoPart` with the lowest price whose description matches the `partDescription` parameter. For example, several of the `AutoPart` elements may have `"headlight"` as their description field. Different headlights will differ in both price and part number. If the `partDescription` parameter is `"headlight"`, then `findCheapest` will return the part number of the cheapest headlight.

```
/* Precondition:  allParts contains at least one element whose
 *                description matches partDescription.
 * Postcondition: Returns the part number of the cheapest AutoPart
 *                whose description matches partDescription.  */
public int findCheapest(String partDescription)
{
    AutoPart part = null;        //AutoPart with lowest price so far
    double min = LARGEVALUE;     //larger than any valid price
    for (AutoPart p : allParts)
    {
        /* more code */
    }
}
```

Which of the following replacements for `/* more code */` will achieve the intended postcondition of the method?

I
```
if (p.getPrice() < min)
{
    min = p.getPrice();
    part = p;
}
return part.getPartNum();
```

II
```
if (p.getDescription().equals(partDescription))
    if (p.getPrice() < min)
    {
        min = p.getPrice();
        part = p;
    }
return part.getPartNum();
```

III
```
if (p.getDescription().equals(partDescription))
    if (p.getPrice() < min)
        return p.getPartNum();
```

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

22. Consider the following method:

```
//Precondition: ob1 and ob2 are distinct Comparable objects.
//Return smaller of ob1 and ob2.
public static Comparable min(Comparable ob1, Comparable ob2)
{
    if (ob1.compareTo(ob2) < 0)
        return ob1;
    else
        return ob2;
}
```

A method in the same class has these declarations:

```
AutoPart p1 = new AutoPart(<suitable values>);
AutoPart p2 = new AutoPart(<suitable values>);
```

Which of the following statements will cause an error?

I
```
System.out.println(min(p1.getDescription(),
p2.getDescription()));
```

II
```
System.out.println(min(((String) p1).getDescription(),
    ((String) p2).getDescription()));
```

III
```
System.out.println(min(p1, p2));
```

(A) None
(B) I only
(C) II only
(D) III only
(E) II and III only

23. This question is based on the following declarations:

```
String strA = "CARROT", strB = "Carrot", strC = "car";
```

Given that all uppercase letters precede all lowercase letters when considering alphabetical order, which is true?

(A) `strA.compareTo(strB) < 0 && strB.compareTo(strC) > 0`
(B) `strC.compareTo(strB) < 0 && strB.compareTo(strA) < 0`
(C) `strB.compareTo(strC) < 0 && strB.compareTo(strA) > 0`
(D) `!(strA.compareTo(strB) == 0) && strB.compareTo(strA) < 0`
(E) `!(strA.compareTo(strB) == 0) && strC.compareTo(strB) < 0`

Questions 24–26 refer to the `ThreeDigitInteger` and `ThreeDigitCode` classes below.

```
public class ThreeDigitInteger
{
    private int myHundredsDigit;
    private int myTensDigit;
    private int myOnesDigit;
    private int myValue;

    //constructor
    //value is a 3-digit int.
    public ThreeDigitInteger(int value)
    { /* implementation not shown */ }

    //Return sum of digits for this ThreeDigitInteger.
    public int digitSum()
    { /* implementation not shown */ }

    //Return sum of hundreds digit and tens digit.
    public int twoDigitSum()
    { /* implementation not shown */ }

    //other methods not shown
        ...
}

public class ThreeDigitCode extends ThreeDigitInteger
{
    private boolean myIsValid;

    //constructor
    //value is a 3-digit int.
    public ThreeDigitCode(int value)
    { /* implementation code */ }

    /* Returns true if ThreeDigitCode is valid, false otherwise.
     * ThreeDigitCode is valid if and only if the remainder when the
     * sum of the hundreds and tens digits is divided by 7 equals the
     * ones digit. Thus 362 is valid while 364 is not.  */
    public boolean isValid()
    { /* implementation not shown */ }
}
```

24. Which is a *true* statement about the classes shown?

(A) The `ThreeDigitInteger` class inherits the `isValid` method from the class `ThreeDigitCode`.

(B) The `ThreeDigitCode` class inherits all of the private instance variables and public accessor methods from the `ThreeDigitInteger` class.

(C) The `ThreeDigitCode` class inherits the constructor from the class `ThreeDigitInteger`.

(D) The `ThreeDigitCode` class can directly access all the private variables of the `ThreeDigitInteger` class.

(E) The `ThreeDigitInteger` class can access the `myIsValid` instance variable of the `ThreeDigitCode` class.

25. Which is correct `/* implementation code */` for the `ThreeDigitCode` constructor?

I
```
super(value);
myIsValid = isValid();
```

II
```
super(value, valid);
```

III
```
super(value);
myIsValid = twoDigitSum() % 7 == myOnesDigit;
```

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

26. Refer to these declarations in a client program:

```
ThreeDigitInteger code = new ThreeDigitCode(127);
ThreeDigitInteger num = new ThreeDigitInteger(456);
```

Which of the following subsequent tests will *not* cause an error?

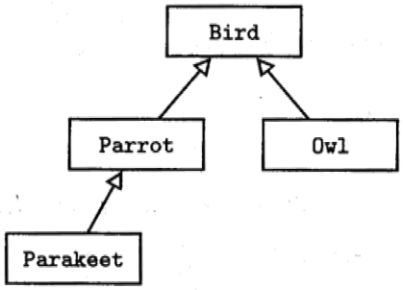
I
```
if (code.isValid())
    ...
```

II
```
if (num.isValid())
    ...
```

III
```
if (((ThreeDigitCode) code).isValid())
    ...
```

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

27. Consider the following hierarchy of classes:



Assuming that each class has a valid default constructor, which of the following declarations in a client program are correct?

I
```
Bird b1 = new Parrot();
Bird b2 = new Parakeet();
Bird b3 = new Owl();
```

II
```
Parakeet p = new Parrot();
Owl o = new Bird();
```

III
```
Parakeet p = new Bird();
```

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

28. Consider an array `arr` and a list `list` that is an `ArrayList<String>`. Both `arr` and `list` are initialized with string values. Which of the following code segments correctly appends all the strings in `arr` to the end of `list`?

I
```
for (String s : arr)
    list.add(s);
```

II
```
for (String s : arr)
    list.add(list.size(), s);
```

III
```
for (int i = 0; i < arr.length; i++)
    list.add(arr[i]);
```

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

29. Refer to the `nextIntInRange` method below:

```
/* Postcondition: Returns a random integer in the range
 *                low to high, inclusive. */
public int nextIntInRange(int low, int high)
{
    return /* expression */
}
```

Which `/* expression */` will always return a value that satisfies the postcondition?

(A) `(int) (Math.random() * high) + low;`

(B) `(int) (Math.random() * (high - low)) + low;`

(C) `(int) (Math.random() * (high - low + 1)) + low;`

(D) `(int) (Math.random() * (high + low)) + low;`

(E) `(int) (Math.random() * (high + low - 1)) + low;`

30. Consider the following `mergeSort` method and the private instance variable `a` both in the same `Sorter` class:

```
private Comparable[] a;

/* Sorts a[first] to a[last] in increasing order using mergesort. */
public void mergeSort(int first, int last)
{
    if (first != last)
    {
        int mid = (first + last) / 2;
        mergeSort(first, mid);
        mergeSort(mid + 1, last);
        merge(first, mid, last);
    }
}
```

Method `mergeSort` calls method `merge`, which has this header:

```
/* Merge a[lb] to a[mi] and a[mi+1] to a[ub].
 * Precondition: a[lb] to a[mi] and a[mi+1] to a[ub] both
 *               sorted in increasing order.  */
private void merge(int lb, int mi, int ub)
```

If the first call to `mergeSort` is `mergeSort(0,3)`, how many *further* calls will there be to `mergeSort` before an array `b[0]...b[3]` is sorted?

(A) 2
(B) 3
(C) 4
(D) 5
(E) 6

31. A programmer has a file of names. She is designing a program that sends junk mail letters to everyone on the list. To make the letters sound personal and friendly, she will extract each person's first name from the name string. She plans to create a parallel file of first names only. For example,

fullName	firstName
Ms. Anjali DeSouza	Anjali
Dr. John Roufaiel	John
Mrs. Mathilda Concia	Mathilda

Here is a method intended to extract the first name from a full name string.

```
/* Precondition:  fullName starts with a title followed by a period.
 *                A single space separates the title, first name,
 *                and last name.
 * Postcondition: Returns the first name only.  */
public static String getFirstName(String fullName)
{
    final String BLANK = " ";
    String temp, firstName;

    /* code to extract first name */

    return firstName;
}
```

Which represents correct /* *code to extract first name* */?

I
```
int k = fullName.indexOf(BLANK);
temp = fullName.substring(k + 1);
k = temp.indexOf(BLANK);
firstName = temp.substring(0, k);
```

II
```
int k = fullName.indexOf(BLANK);
firstName = fullName.substring(k + 1);
k = firstName.indexOf(BLANK);
firstName = firstName.substring(0, k);
```

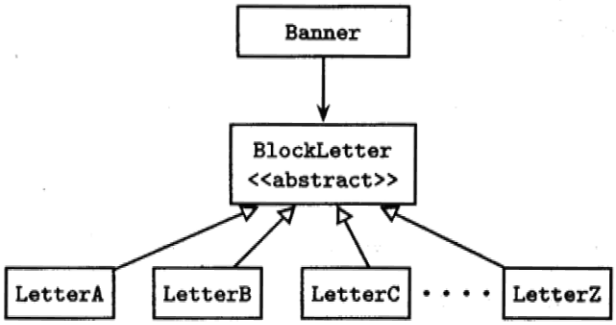
III
```
int firstBlank = fullName.indexOf(BLANK);
int secondBlank = fullName.indexOf(BLANK);
firstName = fullName.substring(firstBlank + 1, secondBlank + 1);
```

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

32. A large hospital maintains a list of patients' records in no particular order. To find the record of a given patient, which represents the most efficient method that will work?
    (A) Do a sequential search on the name field of the records.
    (B) Do a binary search on the name field of the records.
    (C) Use insertion sort to sort the records alphabetically by name; then do a sequential search on the name field of the records.
    (D) Use mergesort to sort the records alphabetically by name; then do a sequential search on the name field of the records.
    (E) Use mergesort to sort the records alphabetically by name; then do a binary search on the name field of the records.

Use the following information for Questions 33 and 34.

Here is a diagram that shows the relationship between some of the classes that will be used in a program to draw a banner with block letters.



The diagram shows that the `Banner` class uses `BlockLetter` objects, and that the `BlockLetter` class has 26 subclasses, representing block letters from A to Z.

The `BlockLetter` class has an abstract `draw` method

```
public abstract void draw();
```

Each of the subclasses shown implements the `draw` method in a unique way to draw its particular letter. The `Banner` class gets an array of `BlockLetter` and has a method to draw all the letters in this array.

Here is a partial implementation of the `Banner` class:

```
public class Banner
{
    private BlockLetter[] letters;
    private int numLetters;

    //constructor. Gets the letters for the Banner.
    public Banner()
    {
        numLetters = <some integer read from user input >
        letters = getLetters();
    }
```

```
    //Return an array of block letters.
    public BlockLetter[] getLetters()
    {
        String letter;
        letters = new BlockLetter[numLetters];
        for (int i = 0; i < numLetters; i++)
        {
            < read in capital letter >

            if (letter.equals("A"))
                letters[i] = new LetterA();
            else if (letter.equals("B"))
                letters[i] = new LetterB();
                    ...          //similar code for C through Y
            else
                letters[i] = new LetterZ();
        }
        return letters;
    }

    //Draw all the letters in the Banner.
    public void drawLetters()
    {
        for (BlockLetter letter : letters)
            letter.draw();
    }

    //Other methods not shown.
        .
}
```

33. You are given the information that `BlockLetter` is an abstract class that is used in the program. Which of the following can you conclude about the class?

    I It *must* have at least one abstract method.

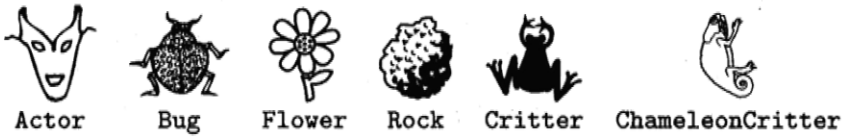
    II It *must* have at least one subclass.

    III No instances of `BlockLetter` can be created.

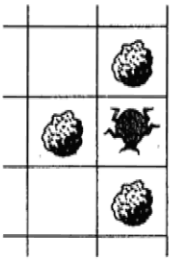
    (A) I only
    (B) II only
    (C) III only
    (D) II and III only
    (E) I, II, and III

34. Which is a *true* statement about method `drawLetters`?
    (A) It is an overloaded method in the `Banner` class.
    (B) It is an overridden method in the `Banner` class.
    (C) It uses polymorphism to draw the correct letters.
    (D) It will cause a compile-time error because `draw` is not implemented in the `BlockLetter` class.
    (E) It will cause a run-time error because `draw` is not implemented in the `BlockLetter` class.

Questions 35–40 involve reasoning about the code from the GridWorld Case Study. A Quick Reference to the case study is provided as part of this exam. The actors in GridWorld are represented in this book with the pictures shown below. Each actor is shown facing north. These pictures almost certainly will be different from those used on the AP exam!

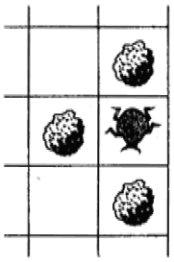


35. Suppose a `Bug` is at the edge of a grid, facing south, as shown.

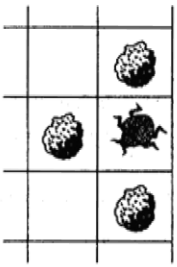


Which of the following correctly represents the state of this part of the grid after the `act` method has been called twice, assuming no other actors enter it?

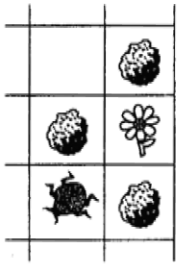
(A)



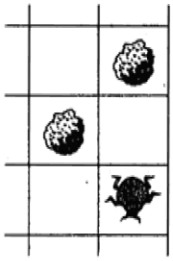
(B)



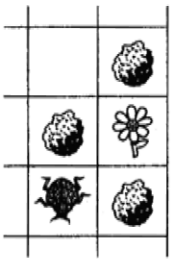
(C)



(D)



(E)



36. Which of the following always makes an actor reverse direction?

I `setDirection(180);`

II `setDirection(getDirection() + Location.HALF_CIRCLE);`

III `setDirection(getDirection() - 180);`

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

37. Here is the implementation of the `act` method in the `Rock` class.

```
public void act()
{ }
```

What would be the effect of omitting this piece of code from the `Rock` class?

I A `Rock` would change location at its turn to act.

II A `Rock` would change direction at its turn to act.

III A `Rock` would change color at its turn to act.

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

38. Which is (are) *true* about the behavior of a `BoxBug` that moves in a grid with many obstacles?

I It will make smaller and smaller squares.

II It will remove itself from the grid when it can no longer move forward.

III It will step on obstacles until it reaches its `sideLength`.

(A) None
(B) I only
(C) II only
(D) I and II only
(E) I and III only

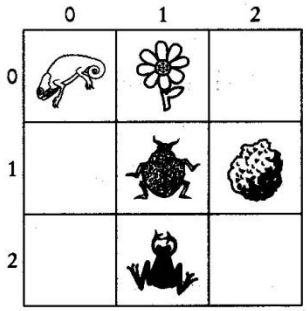
39. Consider a subclass of `Critter` called `CannibalCritter`. A `CannibalCritter` eats only flowers and other critters. Its behavior is otherwise identical to that of a `Critter`. The only method that needs to be overridden in the `CannibalCritter` class is `processActors`. Here is the method.

```
/**
  * Processes the actors.
  * Implemented to "eat" (i.e., remove) all actors that
  * are critters or flowers.
  * @param actors the actors to be processed
  */
public void processActors(ArrayList<Actor> actors)
{
    for (Actor a : actors)
    {
        if ( /* test */ )
            a.removeSelfFromGrid();
    }
}
```

Which replacement for `/* test */` produces the desired behavior for a `CannibalCritter`?

(A) `a instanceof Critter && a instanceof Flower`
(B) `!(a instanceof Critter) && !(a instanceof Flower)`
(C) `a instanceof Critter || a instanceof Flower`
(D) `!(a instanceof Critter) || !(a instanceof Flower)`
(E) `!(a instanceof Critter || a instanceof Flower)`

40. Refer to the bounded grid shown.



Location (0, 0) contains a `ChameleonCritter` facing southwest.
Location (1, 1) contains a `Bug` facing north.
Location (2, 1) contains a `Critter` facing north.
Which is *true* about the actors in the grid?

(A) If it were the `ChameleonCritter`'s turn to act, it could end up in location (0, 1).
(B) If it were the `Rock`'s turn to act, it could end up in location (1, 1).
(C) If it were the `Bug`'s turn to act, it would end up in location (0, 1).
(D) If it were the `Critter`'s turn to act, it could end up in location (1, 2).
(E) If it were the `Flower`'s turn to act, it would end up in location (0, 2).