Practice Test 3

Click here to download a PDF of Practice Test 3.

AP® Computer Science A Exam

SECTION I: Multiple-Choice Questions

DO NOT OPEN THIS BOOKLET UNTIL YOU ARE TOLD TO DO SO.

At a Glance

Total Time
1 hour 30 minutes
Number of Questions
40
Percent of Total Score
50%
Writing Instrument
Pencil required

Instructions

Section I of this examination contains 40 multiple-choice questions. Fill in only the ovals for numbers 1 through 40 on your answer sheet.

Indicate all of your answers to the multiple-choice questions on the answer sheet. No credit will be given for anything written in this exam booklet, but you may use the booklet for notes or scratch work. After you have decided which of the suggested answers is best, completely fill in the corresponding oval on the answer sheet. Give only one answer to each question. If you change an answer, be sure that the previous mark is erased completely. Here is a sample question and answer.

<u>Sample Question</u>

Chicago is a
(A) state
(B) city
(C) country
(D) continent
(E) county

<u>Sample Answer</u>

Ⓐ ● Ⓒ Ⓓ Ⓔ

Use your time effectively, working as quickly as you can without losing accuracy. Do not spend too much time on any one question. Go on to other questions and come back to the ones you have not answered if you have time. It is not expected that everyone will know the answers to all the multiple-choice questions.

About Guessing

Many candidates wonder whether or not to guess the answers to questions about which they are not certain. Multiple-choice scores are based on the number of questions answered correctly. Points are not deducted for incorrect answers, and no points are awarded for unanswered questions. Because points are not deducted for incorrect answers, you are encouraged to answer all multiple-choice questions. On any questions you do not know the answer to, you should eliminate as many choices as you can, and then select the best answer among the remaining choices.

Java Quick Reference

Class Constructors and Methods	**Explanation**
String Class	
`String(String str)`	Constructs a new `String` object that represents the same sequence of characters as `str`
`int length()`	Returns the number of characters in a `String` object
`String substring(int from, int to)`	Returns the substring beginning at index `from` and ending at index `to − 1`
`String substring(int from)`	Returns `substring(from, length())`
`int indexOf(String str)`	Returns the index of the first occurrence of `str`; returns `–1` if not found
`boolean equals(String other)`	Returns `true` if `this` is equal to `other`; returns `false` otherwise
`int compareTo(String other)`	Returns a value <0 if `this` is less than `other`; returns zero if `this` is equal to `other`; returns a value of >0 if `this` is greater than `other`
Integer Class	
`Integer(int value)`	Constructs a new `Integer` object that represents the specified `int` value
`Integer.MIN_VALUE`	The minimum value represented by an `int` or `Integer`

`Integer.MAX_VALUE`	The maximum value represented by an `int` or `Integer`
`int intValue()`	Returns the value of this `Integer` as an `int`
Double Class	
`Double(double value)`	Constructs a new `Double` object that represents the specified `double` value
`double doubleValue()`	Returns the value of this `Double` as a `double`
Math Class	
`static int abs(int x)`	Returns the absolute value of an `int` value
`static double abs(double x)`	Returns the absolute value of a `double` value
`static double pow(double base, double exponent)`	Returns the value of the first parameter raised to the power of the second parameter
`static double sqrt(double x)`	Returns the positive square root of a `double` value
`static double random()`	Returns a `double` value greater than or equal to `0.0` and less than `1.0`
ArrayList Class	
`int size()`	Returns the number of elements in the list
`boolean add(E obj)`	Appends `obj` to end of list; returns `true`
`void add(int index, E obj)`	Inserts `obj` at position index (`0 <= index <= size`), moving elements at position `index` and higher to the right (adds `1` to their indices) and adds 1 to size
`E get(int index)`	Returns the element at position `index` in the list

`E set(int index, E obj)`	Replaces the element at position `index` with `obj`; returns the element formerly at position `index`
`E remove(int index)`	Removes the element at position `index`, moving elements at position `index + 1` and higher to the left (subtracts `1` from their indices) and subtracts 1 from size; returns the element formerly at position `index`
Object Class	
`boolean equals(Object other)`	
`String toString()`	

Section I

COMPUTER SCIENCE A

SECTION I

Time—1 hour and 30 minutes

Number of Questions—40

Percent of total exam grade—50%

Directions: Determine the answer to each of the following questions or incomplete statements, using the available space for any necessary scratchwork. Then decide which is the best of the choices given and fill in the corresponding oval on the answer sheet. No credit will be given for anything written in the examination booklet. Do not spend too much time on any one problem.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Assume that declarations of variables and methods appear within the context of an enclosing class.
- Assume that method calls that are not prefixed with an object or class name and are not shown within a complete class definition appear within the context of an enclosing class.
- Unless otherwise noted in the question, assume that parameters in the method calls are not `null` and that methods are called only when their preconditions are satisfied.

1. Consider the following methods.

```
public void trial()
{
     int a = 10;
     int b = 5;
     doubleValues(a, b);
     System.out.print(b);
     System.out.print(a);
}

public void doubleValues(int c, int d)
{
     c = c * 2;
     d = d * 2;
     System.out.print(c);
     System.out.print(d);
}
```

What is printed as the result of the call `trial()`?

(A) `2010`
(B) `2010105`
(C) `2010510`
(D) `20102010`
(E) `20101020`

2. Consider the following method.

```
/**
 * Precondition: a > b > 0
 */
public static int mystery(int a, int b)
{
     int d = 0;
     for (int c = a; c > b; c--)
     {
          d = d + c;
     }
     return d;
}
```

What is returned by the call `mystery(x, y)`?

(A) The sum of all integers greater than `y` but less than or equal to `x`
(B) The sum of all integers greater than or equal to `y` but less than or equal to `x`
(C) The sum of all integers greater than `y` but less than `x`
(D) The sum of all integers greater than or equal to `y` but less than `x`
(E) The sum of all integers less than `y` but greater than or equal to `x`

3. Consider the following method.

```
public void mystery (int n)
{
      int k;
      for (k = 0; k < n; k++)
      {
            mystery(k);
            System.out.print (k);
      }
}
```

What is printed by the call `mystery(3)` ?

(A) `0123`
(B) `00123`
(C) `0010012`
(D) `00100123`
(E) `001001200100123`

4. Consider an array of integers.

```
4 10 1 2 6 7 3 5
```

If selection sort is used to order the array from smallest to largest values, which of the following represents a possible state of the array at some point during the selection sort process?

(A) `1 4 10  2  3 6  7 5`
(B) `1 2  4  6 10 7  3 5`
(C) `1 2  3 10  6 7  4 5`
(D) `4 3  1  2  6 7 10 5`
(E) `5 3  7  6  2 1 10 4`

5. Consider the following code segment:

```
int k;
int A[];
a = new int [7];
for (k = 0; k < A.length; k++)
{
     A[k] = A.length – k;
}
for (k = 0; k < A.length – 1; k++)
{
     A[k + 1] = A[k];
}
```

What values will `A` contain after the code segment is executed?

(A) `1 1 2 3 4 5 6`
(B) `1 2 3 4 5 6 7`
(C) `6 6 5 4 3 2 1`
(D) `7 7 6 5 4 3 2`
(E) `7 7 7 7 7 7 7`

Questions 6–7 refer to the following two classes.

```
public class PostOffice
{
      // constructor initializes boxes
      // to length 100
```

```
    public PostOffice()
    {    /* implementation not shown */}

    // returns the P.O. box based on the given P.O. box
    number
    // 0 <= theBox < getNumBoxes ()
    public Box getBox(int theBox)
    {    /* implementation not shown */}
    // returns the number of P.O. boxes
    public int getNumBoxes()
    {    /* implementation not shown */}

    // private data members and
    // other methods not shown
}

public class Box
{
    // constructor
    public Box()
    {    /* implementation not shown */}

    // returns the number of this box
    public int getBoxNumber()
    {    /* implementation not shown */}

    // returns the number of pieces
    // of mail in this box
    public int getMailCount()
    {    /* implementation not shown */}
    // returns the given piece of mail
    // 0 <= thePiece < getMailCount ()
    public Mail getMail(int thePiece)
    {    /* implementation not shown */}
    // true if the box has been assigned
    // to a customer
```

```
        public boolean isAssigned()
        {    /* implementation not shown */}
        // true if the box contains mail
        public boolean hasMail()
        {    /* implementation not shown */}
        // private data members and
        // other methods not shown
}
public class Mail
{
        // private members, constructors, and
        // other methods not shown
}
```

6. Consider the following code segment:

```
PostOffice p[];
p = new PostOffice[10];
```

Assuming that the box has been assigned and that it has at least four pieces of mail waiting in it, what is the correct way of getting the fourth piece of mail from the 57th box of the 10th post office of `p`?

(A) `Mail m = p[10].getBox(57).getmail(4);`
(B) `Mail m = p[9].getBox(56).getMail(3);`
(C) `Mail m = p.getMail(57).getMail(4) [10];`
(D) `Mail m = getMail(getBox(p[9], 560, 3);`
(E) `Mail m = new Mail(10, 57, 4);`

7. Consider the incomplete function `printEmptyBoxes` given below. `printEmptyBoxes` should print the box numbers of all of the boxes that have been assigned to a customer but do not contain mail.

```
public void printEmptyBoxes (PostOffice[] p)
```

```
{
     for (int k = 0; k < p.length - 1; k++)
     {
          for (int x = 0; x < p[k].getNumBoxes() - 1; x++)
          {
               /* missing code */
          }
     }
}
```

Which of the following could be used to replace /* missing code */ so that `printBoxesWithoutMail` works as intended?

(A)
```
if (p[k].getBox(x).isAssigned() &&
!p[k].getBox(x).hasMail())
{
  System.out.println(p[k].getBox(x).getBoxNumber());
}
```
(B)
```
if (p[x].getBox(k).isAssigned() &&
!p[x].getBox(k).hasMail())
{
  System.out.println(p[x].getBox(k).getBoxNumber());
}
```
(C)
```
if (p[k].getBox(x).isAssigned() &&
!p[k].getBox(x).hasMail())
{
  System.out.println (p[k].getBoxNumber (x));
}
```
(D)
```
if (p[x].getBox(k).isAssigned() &&
!p[x].getBox (k).hasMail())
{
  System.out.println(p[x].getBoxNumber(k));
}
```
(E)
```
if (p[x].getBox(k).isAssigned() &&
p[x].getBox(k).getMail() == 0)
{
        System.out.println(k);
    }
```

8. Assume that a and b are boolean variables that have been initialized. Consider the following code segment.

```
a = a && b;
b = a || b;
```

Which of the following statements is always true?

I. The final value of a is the same as the initial value of a.
II. The final value of b is the same as the initial value of b.
III. The final value of a is the same as the initial value of b.

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

9. Consider the following code segment.

```
int x;
x = 53;
if (x > 10)
{
     System.out.print("A");
}
if (x > 30)
{
     System.out.print("B");
}
else if (x > 40)
{
     System.out.print("C");
}
```

```
if (x > 50)
{
     System.out.print (“D”);
}
if (x > 70)
{
     System.out.print (“E”);
}
```

What is the output when the code is executed?

(A) A
(B) D
(C) ABD
(D) ABCD
(E) ABCDE

10. Consider the following code segment:

```
int j;
int k;
for (j = −2; j <= 2; j = j + 2)
{
     for (k = j; k < j + 3; k++)
     {
          System.out.print(k + “ ”);
     }
}
```

What is the output when the code is executed?

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

11. Consider the following method.

```
public void mystery (int count, String s)
{
     if (count <= 0)
     {
          return;
     }
     if (count % 3 == 0)
     {
          System.out.print(s + “--” + s);
     }
     else if (count % 3 == 1)
     {
          System.out.print(s + “-” + s);
     }
     else
     {
          System.out.print(s);
     }
     mystery(count - 1, s);
}
```

What is outputted by the call `mystery(5, “X”)`?

(A) XX-XX--XXX-X
(B) XX-XX-XX-XX
(C) XXX--XX-X-XX--XXX
(D) XX-XXX--XXX-XX
(E) XXXXX

Questions 12–13 refer to the following classes and method descriptions.

Class `Table` has a method `getPrice`, which takes no parameters and returns the price of the table.

Class `Chair` also has a method `getPrice`, which takes no parameters and returns the price of the chair.

Class `DiningRoomSet` has a constructor which is passed a `Table` object and an `ArrayList` of `Chair` objects. It stores these parameters in its private data fields `myTable` and `myChairs`.

Class `DiningRoomSet` has a method `getPrice`, which takes no parameters and returns the price of the dining room set. The price of a dining room set is calculated as the sum of the price of its table and all of its chairs.

12. What is the correct way to define the signature of the constructor for the `DiningRoomSet` class?

(A) `public void DiningRoomSet(Table t, ArrayList, chairs)`
(B) `public DiningRoomSet(Table t, ArrayList<Chair> chairs)`
(C) `public void DiningRoomSet(Table t, ArrayList Chair Chairs)`
(D) `public DiningRoomSet(Table t, ArrayList Chair Chairs)`
(E) `public DiningRoomSet(Table t, Chair Chairs)`

13. What is the correct way to implement the `getPrice` method of the `DiningRoomSet` class?

(A)
```
public double getPrice(Table t, ArrayList chairs)
{
  return t.getPrice() + chairs.getPrice();
}
```
(B)
```
public double getPrice(Table t, ArrayList chairs)
{
  return myTable.getPrice() + myChairs.getPrice();
}
```
(C)
```
public double getPrice()
{
     return myTable.getPrice() + myChairs.getPrice();
  }
```

(D)
```
public double getPrice()
  {
     double result = myTable.getPrice();
     for (int k = 0; k < myChairs.size() - 1; k++)
     {
          result += ((Chair)myChairs.get(k)).getPrice();
     }
     return result;
  }
```

(E)
```
public double getPrice()
  {
     double result = myTable.getPrice();
     for (int k = 0; k < myChairs.length - 1; k++)
     {
          result += ((Chair)myChairs[k]).getPrice();
     }
     return result;
  }
```

14. Consider the following output:

```
6  5  4  3  2  1
5  4  3  2  1
4  3  2  1
3  2  1
2  1
1
```

Which of the following code segments produces the above output when executed?

(A)
```
for (int j = 6; j < 0; j--)
  {
     for (int k = j; k > 0; k--)
     {
          System.out.print(k + "  ");
```

```
        }
        System.out.println(“ ”);
    }
```
(B)
```
for (int j = 6; j >= 0; j--)
    {
        for (int k = j; k >= 0; k--)
        {
            System.out.print(k + “ ”);
        }
        System.out.println(“ ”);
    }
```
(C)
```
for (int j = 0; j < 6; j++)
    {
        for (int k = 6 - j; k > 0; k--)
        {
            System.out.print(k + “  ”);
        }
        system.out.println(“ ”);
    }
```
(D)
```
for (int j = 0; j < 6; j++)
    {
        for (int k = 7 - j; k > 0; k--)
        {
            System.out.print(k + “ ”);
        }
        System.out.println(“ ”);
    }
```
(E)
```
for (int j = 0; j < 6; j++)
    {
        for (int k = 6 - j; k >= 0; k--)
        {
            System.out.print(k + “ ”);
        }
        System.out.println(“ ”);
    }
```

15. Consider the following code segment.

```
List<Integer> list = new ArrayList<Integer>();
list.add(new Integer(7));
list.add(new Integer(6));
list.add(1, new Integer(5));
list.add(1, new Integer(4));
list.add(new Integer(3));
list.set(2, new Integer(2));
list.add(1, new Integer(1));
System.out.println(list);
```

What is printed as a result of executing this code segment?

(A) `[1, 4, 2, 7, 6, 3]`
(B) `[7, 1, 4, 2, 6, 3]`
(C) `[7, 2, 5, 4, 3, 1]`
(D) `[7, 6, 2, 4, 3, 1]`
(E) `[7, 1, 2]`

16. Consider the following declarations.

```
public class Animal
{
     String makeSound()
     {
          // Implementation not shown
     }
     String animalType()
     {
          // Implementation not shown
     }
}
public static class Dog extends Animal
{
     public String makeSound(Animal a)
     {
          // Implementation not shown
     }
}
```

Which of the following methods must be included in the declaration of the `Dog` class in order for the class to successfully compile?

I. `public String makeSound()`
II. `public String animalType()`
III. `public String animalType(Animal b)`

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

17. Consider the following two classes.

```
public class Fish
{
     public String endoskeleton = “bone”;

     public void action()
     {
          System.out.println(“splash splash”);
     }
}

public class Shark extends Fish
{
     public void action()
     {
          System.out.println(“chomp chomp”);
     }

     public String endoskeleton = “cartilage”;
}
```

Which of the following is the correct output after the following code segment is executed?

```
Fish Bob = new Shark();
System.out.println(Bob.endoskeleton);
Bob.action();
```

(A) `bone`
    `chomp chomp`

(B) `bone`
    `splash splash`

(C) `cartilage`
    `splash splash`

(D) `cartilage`
    `chomp chomp`

(E) `cartilage`
    `splash splash`
    `chomp chomp`

<u>Questions 18–19</u> refer to the following incomplete method.

The following `insertSort` method sorts the values in an integer array, `sort`, in ascending order.

```
public static void insertSort(int[] sort)
 {
  for (int index = 1; index < sort.length; index++)
  {
  int temp = sort[index];
  while (index > 0 && sort[index - 1] > temp)
    {
  / * missing code * /
    }
  sort[index] = temp;
   }
  }
```

18. Which of the following can be used to replace / * missing code * / so that the `insertSort` method will execute properly?

(A) `sort[index] = sort[index - 1];`
`index++;`
(B) `sort[index - 1] = sort[index];`
`index--;`
(C) `sort[index] = sort[index + 1];`
`index++;`
(D) `sort[index] = sort[index - 1];`
`index--;`
(E) `sort[index] = sort[index + 1];`
`index--;`

19. Assuming that the / * missing code * / is implemented properly, what change can be made to the code in order for the array to be sorted in descending order?

(A) Replace Line 6 with: `while (index < 0 && sort[index - 1] > temp)`
(B) Replace Line 6 with: `while (index < 0 && sort[index - 1] < temp)`
(C) Replace Line 6 with: `while (index > 0 && sort[index - 1] < temp)`
(D) Replace Line 3 with: `for (int index = sort.length - 1; index > 0; index--)`
(E) Replace Line 3 with: `for (int index = 1; index > 0; index--)`

20. Which of the following arrays would be sorted the slowest using insertion sort?

(A) `[3 4 6 2 7 3 9]`
(B) `[3 2 5 4 6 7 9]`
(C) `[9 7 6 5 4 3 2]`
(D) `[2 3 4 5 6 7 9]`

(E) `[9 3 2 4 5 7 6]`

Questions 21–23 refer to the following incomplete class declaration used to represent fractions with integer numerators and denominators.

```
public class Fraction
{
     private int numerator;
     private int denominator;

     public Fraction()
     {
          numerator = 0;
          denominator = 1;
     }

     public Fraction(int n, int d)
     {
          numerator = n;
          denominator = d;
     }

     // postcondition: returns the
     //   numerator
     public int getNumerator()
     {    /* implementation not shown */    }

     // postcondition: returns the
     //   denominator
     public int getDenominator()
     {    /* implementation not shown*/    }

     // postcondition: returns the greatest
     // common divisor of x and y
     public int gcd(int x, int y)
     {  /* implementation not shown*/  }
```

```
        // postcondition: returns the Fraction
        //    that is the result of multiplying
        //    this Fraction and f

        public Fraction multiply(Fraction f)
        {  /* implementation not shown */  }
        //...other methods not shown
}
```

21. Consider the method `multiply` of the `Fraction` class.

```
// postcondition: returns the Fraction
//          that is the result of multiplying
//          this Fraction and f
public Fraction multiply(Fraction f)
{  / * missing code * /  }
```

Which of the following statements can be used to replace / * missing code * / so that the `multiply` method is correctly implemented?

I.
```
return Fraction(
    numerator * f.getNumerator(),
    denominator * f.getDenominator());
```

II.
```
return new Fraction(
    numerator * f.numerator,
    denominator * f.denominator());
```

III.
```
return new Fraction(
    numerator * f.getNumerator(),
    denominator * f.getDenominator());
```

(A) I only
(B) II only
(C) III only

(D) I and III only
(E) II and III only

22. Consider the use of the `Fraction` class to multiply the fractions $\frac{3}{4}$ and $\frac{7}{19}$. Consider the following code:

```
Fraction fractionOne;
Fraction fractionTwo;
Fraction answer;
fractionOne = new Fraction(3, 4);
fractionTwo = new Fraction(7, 19);
/ * missing code * /
```

Which of the following could be used to replace / * missing code * / so that the answer contains the result of multiplying `fractionOne` by `fractionTwo`?

(A) `answer = fractionOne * fractionTwo;`
(B) `answer = multiply(fractionOne, fractionTwo);`
(C) `answer = fractionOne.multiply(fractionTwo);`
(D) `answer = new Fraction(fractionOne, fractionTwo);`
(E)
```
answer = (fractionOne.getNumerator() *
    fractionTwo.getNumerator()) /
    (fractionOne.getDenominator() *
    fractionTwo.getDenominator());
```

23. The following incomplete class declaration is intended to extend the `Fraction` class so that fractions can be manipulated in reduced form (lowest terms).

Note that a fraction can be reduced to lowest terms by dividing both the numerator and denominator by the greatest common divisor (gcd) of the numerator and denominator.

```
public class ReducedFraction extends Fraction
{
     private int reducedNumerator;
     private int reducedDenominator;
     //…constructors and other methods not shown
}
```

Consider the following proposed constructors for the `ReducedFraction` class:

I.
```
public ReducedFraction( )
   {
     reducedNumerator = 0;
     reducedDenominator = 1;
   }
```
II.
```
public ReducedFraction(int n, int d)
   {
     numerator = n;
     denominator = d;
     reducedNumerator = n / gcd(n, d);
     reducedDenominator = d / gcd(n, d);
   }
```
III.
```
public ReducedFraction(int n, int d)
   {
     super(n, d);
     reducedNumerator = n / gcd(n, d);
     reducedDenominator = d / gcd(n, d);
   }
```

Which of these constructor(s) would be legal for the `ReducedFraction` class?

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

(E) II and III only

24. Consider `s1` and `s2` defined as follows.

```
String s1 = new String(“hello”);
String s2 = new String(“hello”);
```

Which of the following is/are correct ways to see whether `s1` and `s2` hold identical strings?

I. `if (s1 == s2)`
    /** s1 and s2 are identical */
II. `if (s1.equals(s2))`
    /** s1 and s2 are identical */
III. `if (s1.compareTo(s2) == 0)`
    /** s1 and s2 are identical */

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

25. Consider the following variable and method declarations:

```
String s;
String t;
public void mystery (String a, String b)
{
     a = a + b;
     b = b + a;
}
```

Assume that `s` has the value “`Elizabeth`” and `t` has the value “`Andrew`” and `mystery (s, t)` is called. What are the values of `s` and `t` after the call to `mystery`?

(A) `s: Elizabeth, t: Andrew`
(B) `s: ElizabethAndrew, t: AndrewElizabeth`
(C) `s: ElizabethAndrew, t: AndrewElizabethAndrew`
(D) `s: ElizabethAndrew, t: ElizabethAndrewAndrew`
(E) `s: ElizabethAndrewElizabeth, t: AndrewElizabethAndrew`

26. Consider the following incomplete and *incorrect* class and interface declarations:

```
public class ComparableObject
{
    public int compareTo(Object o)
    {
        //method body not shown
    }
//other methods and variables not shown
}
public class Point extends ComparableObject
{
    private int x;
    private int y;
    public boolean compareTo(Point other)
    {
        return (x == other.x &&
            y == other.y);
    }
    //...constructors and other methods
    //  not shown
}
```

For which of the following reasons, if any, is the above class structure incorrect?

I. Objects may not access private data fields of other objects in the same class.

II. The `ComparableObject` class requires that `compareTo` be passed as an `Object` rather than a `Point`.
III. The `ComparableObject` class requires that `compareTo` return an `int` rather than a `boolean`.

(A) I only
(B) III only
(C) I and III only
(D) II and III only
(E) None, the above class declarations are correct.

27. Consider the following abstraction of a `for` loop where <1>, <2>, <3>, and <4> represent legal code in the indicated locations:

```
for (<1>; <2>; <3>)
{
     <4>
}
```

Which of the following `while` loops has the same functionality as the above `for` loop?

(A)
```
<1>;
while (<2>)
{
   <3>;
   <4>
}
```
(B)
```
<1>;
while (<2>)
{
   <4>
   <3>;
}
```

(C) <1>;
```
    while (!<2>)
{
    <3>;
    <4>
}
```
(D) <1>;
```
while (! <2>)
{
    <4>
    <3>;
}
```
(E) <1>;
```
<3>;
while (<2>)
{
    <4>
    <3>;
}
```

28. Consider the following expression:

```
a / b + c − d % e ∗ f
```

Which of the expressions given below is equivalent to the one given above?

(A) `((a / b) + (c − d)) % (e ∗ f)`
(B) `((((a / b) + c) − d) % e) ∗ f`
(C) `((a / b) + c) − (d % (e ∗ f)`
(D) `(a / ((b + c) − d) % e) ∗ f`
(E) `((a / b) + c) − ((d % e) ∗ f)`

29. Assume that a program declares and initializes x as follows:

```
String[] x ;
x = new String[10] ;
initialize(x);     // Fills the array x with
                   // valid strings each of
                   // length 5
```

Which of the following code segments correctly traverses the array and prints out the first character of all ten strings followed by the second character of all ten strings, and so on?

I.
```
int i;
int j;
for (i = 0; i < 10; i++)
   for (j = 0; j < 5; j++)
        System.out.print(x[i].substring(j, j + 1));
```
II.
```
int i;
int j;
for (i = 0; i < 5; i++)
   for (j = 0; j < 10; j++)
        System.out.print(x[j].substring(i, i + 1));
```
III.
```
int i;
int j;
for (i = 0; i < 5; i++)
   for (j = 0; j < 10; j++)
        System.out.print(x[i].substring(j, j + 1));
```

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

30. Consider the following declaration and assignment statements:

```
int a = 7;
int b = 4;
double c;
c = a / b;
```

After the assignment statement is executed, what’s the value of `c?`

(A) 1.0
(B) 1.75
(C) 2.0
(D) An error occurs because `c` was not initialized.
(E) An error occurs because `a` and `b` are integers and `c` is a double.

31. Consider the following code segment:

```
int x;
x = /* initialized to an integer */
if (x % 2 == 0 && x / 3 == 1)
     System.out.print(“Yes”);
```

For what values of `x` will the word `“Yes”` be printed when the code segment is executed?

(A) 0
(B) 4
(C) Whenever `x` is even and `x` is not divisible by 3
(D) Whenever `x` is odd and `x` is divisible by 3
(E) Whenever `x` is even and `x` is divisible by 3

32. Consider the following incomplete class definition:

```
public class SomeClass
{
     private String myName;
```

```
        // postcondition: returns myName
        public String getName( )
        {/* implementation not shown */ }
        // postcondition: myName == name
        public void setName(String name)
        { /* implementation not shown */ }
        //…constructors, other methods,
        //   and private data not shown
}
```

Now consider the method `swap`, not part of the `SomeClass` class.

```
// precondition: x and y are correctly
//   constructed
// postcondition: the names of objects
//   x and y are swapped
public void swap (SomeClass x, SomeClass y)
{
        / * missing code * /
}
```

Which of the following code segments can replace / * missing code * / so that the method `swap` works as intended?

I.
```
SomeClass temp;
temp = x;
x = y;
y = temp;
```

II.
```
String temp;
temp = x.myName;
x.myName = y.myName;
y.myName = temp;
```

III.
```
String temp;
```

```
temp = x.getName();
x.setName(y.getName());
y.setName(temp);
```

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

33. A bookstore wants to store information about the different types of books it sells.

For each book, it wants to keep track of the title of the book, the author of the book, and whether the book is a work of fiction or nonfiction.

If the book is a work of fiction, then the bookstore wants to keep track of whether it is a romance novel, a mystery novel, or science fiction.

If the book is a work of nonfiction, then the bookstore wants to keep track of whether it is a biography, a cookbook, or a self-help book.

Which of the following is the best design?

(A) Use one class, `Book`, which has three data fields: `String title`, `String author`, and `int bookType`.
(B) Use four unrelated classes: `Book`, `Title`, `Author`, and `BookType`.
(C) Use a class `Book` which has two data fields: `String title`, `String author`, and a subclass: `BookType`.

(D) Use a class `Book` which has two data fields: `String title`, `String author`, and six subclasses: `RomanceNovel`, `Mystery`, `ScienceFiction`, `Biography`, `Cookbook`, and `SelfHelpBook`.

(E) Use a class `Book` which has two data fields: `String title`, `String author`, and two subclasses: `FictionWork` and `NonFictionWork`. The class `FictionWork` has three subclasses, `RomanceNovel`, `Mystery`, and `ScienceFiction`. The class `NonFictionWork` has three subclasses: `Biography`, `Cookbook`, and `SelfHelpBook`.

34. Consider the following code:

```
public int mystery(int x)
{
      if (x == 1)
            return <missing value>;
      else
            return(2 * mystery(x - 1)) + x;
}
```

Which of the following can be used to replace <missing value> so that `mystery (4)` returns `34` ?

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

35. Consider the following code segment:

```
int [ ] X;
int [ ] Y;
int k;
X = initializeX();          // returns a valid
```

```
                     // initialized int [ ]
Y = initializeY();        // returns a valid
                     // initialized int [ ]
for (k = 0;
     k < X.length && X[k] == Y[k];
     k++)
{

     /* some code */
}
```

Assuming that after `X` and `Y` are initialized, `X.length == Y.length`, which of the following must be true after executing this code segment?

(A) `k < X.length`
(B) `k < X.length && X[k] == Y[k]`
(C) `k < X.length && X[k] != Y[k]`
(D) `k >= X.length || X[k] == Y[k]`
(E) `k >= X.length || X[k] != Y[k]`

36. Which of the following would NOT cause a run-time exception?

(A) Dividing an integer by zero
(B) Using an object that has been declared but not instantiated
(C) Accessing an array element with an array index that is equal to the length of the array
(D) Attempting to create a substring beginning at a negative index
(E) Attempting to call a method with the wrong number of arguments

37. Assume that `a` and `b` are properly initialized variables of type `Double`.

Which of the following is an equivalent expression to:

```
a.doubleValue() != b.doubleValue()
```

(A) `a != b`
(B) `a.notEquals(b)`
(C) `!(a.doubleValue() .equals(b.doubleValue()))`
(D) `!(a.compareTo(b))`
(E) `a.compareTo(b) != 0`

38. Which of the following would be the LEAST effective way of ensuring reliability in a program?

(A) Encapsulating functionality in a class by declaring all data fields to be public
(B) Defining and following preconditions and postconditions for every method
(C) Including assertions at key places in the code
(D) Using descriptive variable names
(E) Indenting code in a consistent and logical manner

39. Consider a dictionary that has 1,024 pages with 50 words on each page.

In order to look up a given target word, a student is considering using one of the following three methods:

Method 1

Use a binary search technique to find the correct page (comparing the target word with the first word on a given page). When the correct page is found, use a sequential search technique to find the target word on the page.

Method 2

Use a sequential search technique to find the correct page (comparing the target word with the first word on a given page). When the correct page is found, use another sequential search technique to find the target word on the page.

Method 3

Use a sequential search technique on all of the words in the dictionary to find the target word.

Which of the following best characterizes the greatest number of words that will be examined using each method?

(A) Method 1: 10, Method 2: 50, Method 3: 1,024
(B) Method 1: 55, Method 2: 512, Method 3: 2,560
(C) Method 1: 55, Method 2: 537, Method 3: 25,600
(D) Method 1: 60, Method 2: 1,074, Method 3: 1,074
(E) Method 1: 60, Method 2: 1,074, Method 3: 51,200

40. Consider the following recursive method.

```
public static int mystery(int m)
{
     if (m == 0)
     {
          return 0;
     }
     else
     {
          return 4 + mystery(m – 2);
     }
}
```

Assuming that j is a positive integer and that m = 2j, what value is returned as a result of the call `mystery(m)`?

(A) `0`

(B) `m`
(C) `2m`
(D) `j`
(E) `2j`

END OF SECTION I

IF YOU FINISH BEFORE TIME IS CALLED, YOU MAY CHECK YOUR WORK ON THIS SECTION.

DO NOT GO ON TO SECTION II UNTIL YOU ARE TOLD TO DO SO.

COMPUTER SCIENCE A

SECTION II

Time—1 hour and 30 minutes

Number of Questions—4

Percent of Total Grade—50%

Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA™.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

FREE-RESPONSE QUESTIONS

1. A day care has a program to keep track of its employees and which children they teach during the day. An `Employee` has a minimum and maximum age they can teach. The `DayCare` also has a maximum ratio that specifies the maximum number of children a single employee can teach. Below is the full `DayCare` class:

```
public class DayCare
{
     private ArrayList<Employee> employees;
     private ArrayList<Child> children;
     private int maxRatio;

     public DayCare(int maxRatio)
     {
          employees = new ArrayList<Employee>();
          children = new ArrayList<Child>();
          this.maxRatio = maxRatio;
     }

     public boolean findEmployeeForChild(Child c)
     {
          /* To be completed in part (a) */
     }

     public boolean runDayCare()
     {
          /* To be completed in part (b) */
     }

     public boolean addChild(Child c)
     {
          /* To be completed in part (c) */
     }
}
```

(a) An `Employee` can only teach children between the employee's minimum age (inclusive) and maximum age (inclusive). They can also only teach children up to the day care's maximum ratio (inclusive). Below is the full `Employee` class.

```
public class Employee
{
     /* Instance variables not shown */

     public Employee(String name, String id, int min, int
     max)
```

```
    {
        /* Implementation not shown */
    }

    // Return the number of children currently assigned to this
    Employee
    public int childrenAssigned()
    {
        /* Implementation not shown */
    }

    // Assign a new child to this Employee
    public void assignChild(Child c)
    {
        /* Implementation not shown */
    }

    // Determine whether this Employee can teach a Child
    based on the child's age
    public boolean canTeach(int age)
    {
        /* Implementation not shown */
    }
}
```

A `Child` has accessors to get their name and age. While the implementation of `Child` is not shown, you can assume the accessors are called `getName` and `getAge`.

Complete the `findEmployeeForChild` method below that assigns a `Child` to the first `Employee` who can teach the `Child` and who has not reached the maximum ratio of the `DayCare`.

```
/* Return true if an Employee was found for the Child, false
otherwise */
public boolean findEmployeeForChild(Child c)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

(b) In order for the `DayCare` to run for a day, each `Child` must be assigned an `Employee`. If an `Employee` cannot be found for a `Child`, the `DayCare` cannot run for the day.

Complete the `runDayCare` method below that finds an `Employee` for each `Child` in the `children ArrayList`.

/* Return `true` if an `Employee` was found for each `Child, false` otherwise */

```
public boolean runDayCare()
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

(c) When a `Child` is added to the roster of the `DayCare`, the `DayCare` should first make sure there is an `Employee` available to teach that `Child`.

Complete the `addChild` method below that adds a `Child` to the `children ArrayList` if an `Employee` is available to teach that `Child`.

/* Return `true` if the `Child` was added to the `ArrayList, false` otherwise */

```
public boolean addChild(Child c)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

2. A baseball team consists of different people including players, coaches, and people who work in the front office making trades and other transactions. The following `Person` class is used for all of the people who work for the team.

   Each person has a name and an age.

```
public class Person
{
     private String fullName;
     private int age;

     public Person(String s, int a)
     {
          fullName = s;
          age = a;
     }

     // Accessors for name and age
     public String getName()
     {
          return fullName;
     }

     public int getAge()
     {
            return age;
        }
    }
```

A `Player` has a name and age just like any person on the team, but also has a `position`. The position could be something like "catcher," "left fielder," or "infielder." Players should also be able to change their positions using a method called `changePosition`. Here is an example of a `Player` object:

```
        Player p = new Player("Sammy Sosa", 32, "right
        fielder");
        p.changePosition("outfielder");
```

Write the entire `Player` class.

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

3. A class is designed to store someone's full name. You can assume the name has only one space between the first name and the last name. The class has methods to extract the first name, last name, and number of vowels in the name. You can see an example below.

```
String fullName = "Katherine Johnson";
Name.getFirstName(fullName); // Returns "Katherine"
Name.getLastName(fullName); // Returns "Johnson"
Name.countVowels(fullName); // Returns 6
```

(a) The `getFirstName` method returns the first name based on a given full name. You can assume that `fullName` has only one space between the first name and the last name. Write the `getFirstName` method.

```
public static String getFirstName(String name)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

(b) The `getLastName` method returns the last name based on a given full name. You can assume that `fullName` has only one space between the first name and the last name. Write the `getLastName` method.

```
public static String getLastName(String name)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

(c) The `countVowels` method counts the number of vowels in the given full name. You can assume we will count only the letters `a`, `e, i, o`, and `u` as vowels. Write the entire `countVowels` method.

```
public static int countVowels(String name)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

4. A city parking lot has a sign that keeps track of how many parking spaces are available in the lot. The class for the parking lot is detailed below.

```
public class ParkingLot
{
     private Car[ ][ ] lot;

     public ParkingLot(int rows, int cols)
     {
          lot = new Car[rows][cols];
     }

     public int openSpaces()
     {
          // Complete in part (a)
     }

     public boolean parkCar(Car newCar)
     {
          // Complete in part (b)
     }
}
```

(a) Write the `openSpaces` method that returns the number of spaces available in the parking lot. If a space is empty, it will be equal to `null`.

/* Return the number of empty spaces in the parking lot */

```
public int openSpaces( )
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

(b) Complete the `parkCar` method that puts a new car in any space in the parking lot and returns `true` if it was able to do so. It should return `false` if there are no empty spaces. You should use the `openSpaces` method to receive full credit.

/* Return `true` if there is an open spot to park the `newCar, false` otherwise. The car should be added to the `lot` 2D array if there is an open spot. */

```
public boolean parkCar(Car newCar)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

STOP

END OF EXAM

Practice Test 3: Answers and Explanations

PRACTICE TEST 3 ANSWER KEY

1.	C	21.	C
2.	A	22.	C
3.	C	23.	D
4.	C	24.	D
5.	E	25.	A
6.	B	26.	E
7.	A	27.	B
8.	B	28.	E
9.	C	29.	B
10.	E	30.	A
11.	A	31.	B
12.	B	32.	B
13.	D	33.	E
14.	C	34.	C
15.	B	35.	E
16.	E	36.	E
17.	A	37.	E
18.	D	38.	A
19.	C	39.	E
20.	C	40.	C

ANSWER EXPLANATIONS

Section I: Multiple-Choice Questions

1. **C**

This question tests how well you understand assigning values to variables and following the steps of the code. When `trial()` is called, `a` is assigned to the integer value of 10 and `b` is assigned to the integer value of 5. Next, `doubleValues()` is called with `a` and `b` as inputs `c` and `d`. The method `doubleValues()` multiplies the input values `c` and `d` by 2 and prints them out. Thus, the value of `c` is $10 * 2 = 20$ and the value of `d` is $5 * 2 = 10$. Because `System.out.print()` does not print any line breaks or spaces, the resulting print is `2010`. While `c` and `d` have been reassigned new values, these values exist only within the `doubleValues()` call, so the values of `a` and `b` are unchanged. After the `doubleValues()` method is completed, the `trial()` method then prints the values of `b` and then `a`, printing `510`. Once again, no spaces or line breaks are printed, so, combined together, what is printed from calling `trial()` is `2010510`. Therefore, the correct answer is (C).

2. **A**

The method `mystery(int a, int b)` takes as input integers `a` and `b` with the precondition that `a` > `b` > 0. Plug in `x` = 5 and `y` = 2. These values are passed as parameters to make `a` = 5 and `b` = 2. Set `d = 0`. Now go to the `for` loop, initializing `c` = `a` = 5. Since it is still the case that `c` > `b`, execute the `for` loop. Execute `d = d + c` by setting `d` = $0 + 5 = 5$. Decrease `c` by 1 to get `c` = 4. Since `c` > `b`, execute the `for` loop again. Execute `d = d + c` by setting `d` = $5 + 4 = 9$. Decrease `c` by 1 to get `c` = 3. Since it is still the case that `c` > `b`, execute the `for` loop again. Execute `d = d + c` by setting `d` =

9 + 3 = 12. Decrease `c` by 1 to get `c` = 2. Since it is no longer the case that `c > b`, stop executing the `for` loop. Return `d` = 12. Now go through the choices and eliminate any that don't describe a return of 12. Choice (A) is the sum of all integers greater than `y` but less than or equal to `x`. The sum of all integers greater than 2 but less than or equal to 5 is 3 + 4 + 5 = 12, so keep (A). Choice (B) is the sum of all integers greater than or equal to `y` but less than or equal to `x`. The sum of all integers greater than or equal to 2 but less than or equal to 5 is 2 + 3 + 4 + 5 = 14, so eliminate (B). Choice (C) is the sum of all integers greater than `y` but less than `x`. The sum of all integers greater than 2 but less than 5 is 3 + 4 = 7, so eliminate (C). Choice (D) is the sum of all integers greater than or equal to `y` but less than `x`. The sum of all integers greater than or equal to 2 but less than 5 is 2 + 3 + 4 = 9, so eliminate (D). Choice (E) is the sum of all integers less than `y` but greater than or equal to `x`. However, there are no integers that are both less than 2 and greater than or equal to 5, so eliminate (E). Only one answer remains. The correct answer is (A).

3. **C**

The question asks for what is printed by the call `mystery(3)`. In the call, 3 is taken as a parameter and assigned to `n`. The integer `k` is then initialized in the `for` loop and set equal to 0. Since `k < n`, execute the `for` loop. Call `mystery(0)`, while keeping your place in the call of `mystery(3)`. Again, `k` is initialized in the `for` loop and set equal to 0. In this call, since `n` = 0, it is not the case that `k < n`, so do not execute the `for` loop. This completes the call of `mystery(0)`, so return to `mystery(3)`. The next command is `System.out.print(k)`. Since `k` = 0, the first character printed is `0`. Look to see whether any choices can be eliminated. All choices begin with `0`, so no choices can be eliminated. Continue with the method call. Since this is the last command of the `for` loop, increment `k` to get `k` = 1. Since it is still the case that `k` < `n`, execute the `for` loop again. Call `mystery(1)`, again keeping your place in the call of `mystery(3)`. Again, `k` is initialized in the `for` loop and set equal to 0. In this call, since `n` = 1, it is the case that `k` < `n`, so execute the `for` loop. Call `mystery(k)`, which is `mystery(0)`. As seen above, `mystery(0)` prints nothing, so execute the next line in `mystery(1)`, which is `System.out.print(k)`. Since `k` = 0, print `0`. Eliminate (A), since the second character printed is not `0`. Increment `k` to get `k` = 1. Since it is no longer the case that `k` < `n`, do not execute the `for` loop again and end this call of `mystery(1)`. Return to the original call of `mystery(3)`, where `k` = 1 and `System.out.print(k)` is the next statement. Print `1`. All remaining choices have `1` as the third character, so do not eliminate any choices. This is the end of the `for` loop, so increment `k` to get `k` = 2. Since `n` = 3, it is still the case that `k` < `n`, so execute the `for` loop. Call `mystery(2)` while holding your place in `mystery(3)`. Again, `k` is initialized in the `for` loop and set equal to 0. In this call, since `n` = 2, it is the case that `k` < `n`, so execute the `for` loop. Call `mystery(0)`, which prints nothing. Execute `System.out.print(k)` to print `0`. Eliminate (B), since the next character is not `0`. Increment `k` to get `k` = 1. Since `k` < `n`, execute the `for` loop. Call `mystery(1)`, which, as described above, prints `0`. All remaining choices have `0` as the next character, so don't eliminate any choices. The next command in `mystery(2)` is `System.out. print(k)`, so print `1`. Again, keep all the remaining choices. In `mystery(2)`, increment `k` to get `k` = 2. Since `n` = 2, it is no longer the case that `k` < `n`, so stop executing the `for` loop. End `mystery(2)` and return to `mystery(3)`, where `k` = 2 and the next command is `System.out. print(k)`, so print `2`. Again, keep all remaining choices. Increment `k` to get `k` = 3. Since it is no longer the case that `k` < `n`, stop executing the `for` loop. The method call is complete for `mystery(3)`, so there will be no more printed characters. Eliminate (D) and (E), which have further characters printed.

In other words, the calling sequence looks like this

```
Function Call                                  Prints
mystery(3)
   mystery(0)
   System.out.print(0)                         0
   mystery(1)
      mystery(0)
      System.out.print(0)                      0
   System.out.print(1)                         1
   mystery(2)
      mystery(0)
      System.out.print(0)                      0
      mystery(1)
         mystery(0)
         System.out.print(0)                   0
      System.out.print(1)                      1
   System.out.print(2)                         2
```

The correct answer is (C).

4. **C**

`SelectionSort` walks through the array to find the smallest element in the part of the array not yet sorted. It then swaps that smallest element with the first unsorted element.

Start with the original array of values.

```
4  10  1  2  6  7  3  5
```

The smallest element is 1. Swap that with the first unsorted element, 4. The array now looks like this.

```
1   10   4   2   6   7   3   5
```

This is not a choice, so continue. The smallest element in the unsorted part of the array (from 10 to 5) is 2. Swap that with the first unsorted element, 10.

```
1   2   4   10   6   7   3   5
```

This is still not a choice, so continue this process. The smallest element in the unsorted part of the array (from 4 to 5) is 3. Swap that with the first unsorted element, 4.

```
1   2   3   10   6   7   4   5
```

This is (C), so stop here. If you're interested, here are the complete moves for selection sort.

```
1   2   3   4   6   7   10   5

1   2   3   4   5   7   10   6

1   2   3   4   5   6   10   7

1   2   3   4   5   6   7   10
```

The correct answer is (C).

5. **E**

Execute the commands. The integer `k` and the integer array `A` are initialized. The array `A` is given length 7. Thus, the array, `A`, has seven elements with indexes from 0 to 6. Execute the first `for` loop. Set `k` = 0. Since `k` < `A.length`, execute the `for` loop, which sets `A[0]` equal to `A.length - k` = $7 - 0 = 7$. Increment `k` to get `k` = 1. Since $1 < 7$, execute the `for` loop, which sets `A[1]` equal to `A.length - k` = $7 - 1 = 6$. Increment `k` to get `k` = 2. Since

2 < 7, execute the `for` loop, which sets `A[2]` equal to `A.length –` `k` = 7 – 2 = 5. Increment `k` to get `k` = 3. Since 3 < 7, execute the `for` loop, which sets `A[3]` equal to `A.length – k` = 7 – 3 = 4. Increment `k` to get `k` = 4. Since 4 < 7, execute the `for` loop, which sets `A[4]` equal to `A.length – k` = 7 – 4 = 3. Increment `k` to get `k` = 5. Since 5 < 7, execute the `for` loop, which sets `A[5]` equal to `A.length – k` = 7 – 5 = 2. Increment `k` to get `k` = 6. Since 6 < 7, execute the `for` loop, which sets `A[6]` equal to `A.length – k` = 7 – 6 = 1. Increment `k` to get `k` = 7. Since it is not the case that 7 < 7, stop executing the `for` loop. Therefore, the array has the values

```
7  6  5  4  3  2  1
```

The second `for` loop puts the value of the `k`th element of the array into the `(k + 1)`st element. Be careful!

A quick glance might lead you to believe that each value is shifted in the array one spot to the right, giving an answer of (D). However, a step-by-step analysis demonstrates that this will not be the case. The `for` loop sets `k` = 0. Since 0 < `A.length` – 1, execute the `for` loop. The command in the `for` loop is `A[k + 1] = A[k]`. Since `k` = 0, set `A[1] = A[0] =` 7. Increment `k` to get `k` = 1. Since it is still the case that `k < A.length` – 1, execute the `for` loop. Since `k` = 1, set `A[2] = A[1] =` 7. Continue in this manner. Since `A.length` – 1 = 7 – 1 = 6, only execute the `for` loop through `k` = 5. Each execution of the `for` loop is shown below.

k	Assignment	A[]
0	A[1] = A[0]	7 7 5 4 3 2 1
1	A[2] = A[1]	7 7 7 4 3 2 1
3	A[3] = A[2]	7 7 7 7 3 2 1
4	A[4] = A[3]	7 7 7 7 7 2 1

```
 5      A[5] = A[4]     7 7 7 7 7 7 1
        A[6] = A[5]     7 7 7 7 7 7 7
```

The final values contained by `A` are `7 7 7 7 7 7 7`. The correct answer is (E).

6. **B**

Choices (C), (D), and (E) are syntactically invalid according to the given class definitions. In (C), `p` is used as a `PostOffice` object rather than an array of `PostOffice` objects. Choice (D) treats `getMail` and `getBox` as static methods without invoking them from an object. Choice (E) creates a new `Mail` object attempting to use a `Mail` constructor. Even if such a constructor were available, there would be no way for the constructor to know about `p`, the array of `PostOffices`.

Choices (A) and (B) differ only in the indexes of the array and methods. There are two clues in the question that indicate that (B) is the correct answer. First, `p` is declared as an array of 10 `Postoffices`. This means that `p[10]` would raise an `ArrayListOutOfBoundsException`. Remember that `p[9]` actually refers to the 10th post office, since array indexes begin with 0. Second, the comments in the class definitions for the `getBox` and `getMail` methods indicate that the parameter they take is zero-based. Therefore, they should be passed an integer one less than the number of the box or piece of mail needed. For either reason, eliminate (A). The correct answer is (B).

7. **A**

In the method `printEmptyBoxes`, the loop variable `k` refers to the index of the post office and the loop variable `x` refers to the index of the box within the post office. Choice (A) is correct. It checks to see whether the box is assigned and whether it does not have mail using the appropriate methods of the `Box` class. It then prints out the box number of the box. Choice (B) is similar to (A) but incorrectly interchanges `x` and `k`. Eliminate (B). Choice (C) omits the call to the method `getBox`. Therefore, it attempts to call the `getBoxNumber()` method of the `PostOffice` object `p[k]`. Since `PostOffice` objects do not have a `getBoxNumber()` method, this would result in a compile error. Eliminate (C). Choice (D) is similar to (C), but interchanges `x` and `k`. Eliminate (D). Choice (E) prints the value of k, which is the index of the post office rather than the index of the box. Eliminate (E). The correct answer is (A).

8. **B**

There are four possibilities for the values of `a` and `b`. Either both `a` and `b` are true, both `a` and `b` are false, `a` is true and `b` is false, or `a` is false and `b` is true.

Analyze the possibilities in a table.

a initial value)	**b (initial value)**	**a = a && b (final value)**	**b = a || b (final value)**
`true`	`true`	`true`	`true`
`true`	`false`	`false`	`false`
`false`	`true`	`false`	`true`
`false`	`false`	`false`	`false`

Remember when calculating `b = a || b` that `a` has already been modified. Therefore, use the final, rather than the initial, value of `a` when calculating the final value of `b`.

Go through each statement. Statement I says that the final value of `a` is equal to the initial value of `a`. This is not the case in the second row of the table, so Statement I is not always true. Eliminate any choice that includes it: (A) and (D). Statement II says that the final value of `b` is equal to the initial value of `b`. This is true in each row and, thus, Statement II is always true, so eliminate the choice that does not include it: (C). Statement III says that the final value of `a` is equal to the initial value of `b`. This is not the case in the third row of the table, so Statement III is not always true. Eliminate any choice that includes it: (E). The correct answer is (B).

9. **C**

The best way to solve this problem is to look at each `if` statement individually. The integer `x` is initialized and then set equal to 53. The next statement is an `if` statement with the condition `(x > 10)`. Since $53 > 10$, execute the statement, `System.out.print("A")`, and print `A`. The next statement is an `if-else` statement, this one having the condition `(x > 30)`. Since $53 > 30$, execute the `if` statement, `System.out.print("B")`, and print `B`. Even though $53 > 40$, because the next part of the statement is an `else` statement, it cannot be executed if the original `if` statement was executed. Therefore, do not execute the `else` statement, and do not print `C`. The next statement is another `if` statement, this one having the condition `(x > 50)`. Since $53 > 50$, execute the statement, `System.out.print("D")`, and print `D`. The next statement is another `if` statement, this one having the condition `(x > 70)`. Since it is not the case that $53 > 70$, do not execute the statement. There is no further code to execute, so the result of the printing is `ABD`. The correct answer is (C).

10. **E**

This problem tests your ability to work with nested `for` loops. Although it appears complicated at first glance, it can easily be solved by systematically walking through the code.

Be sure to write the values of the variables `j` and `k` down on paper; don't try to keep track of `j` and `k` in your head. Use the empty space in the question booklet for this purpose.

Code	**j**	**k**	**Output**
Set `j` to the value `–2.`	–2		
Test the condition: `j <= 2?` Yes.	–2		
Set `k` to the value that `j` has.	–2	–2	
Test the condition: `k < j + 3?` Yes.	–2	–2	
Output `k.`	–2	–2	`–2`
Update k: `k++`	–2	–1	`–2`
Test the condition: `k < j + 3?` Yes.	–2	–1	`–2`
Output `k.`	–2	–1	`–2 –1`
Update k: `k++`	–2	0	`–2 –1`
Test the condition: `k < j + 3?` Yes.	–2	0	`–2 –1`
Output `k.`	–2	0	`–2 –1 0`
Update k: `k++`	–2	1	`–2 –1 0`

Test the condition: `k < j + 3`? No.	–2	1	`–2 –1 0`
Update j: `j` = `j` + 2	0	1	`–2 –1 0`
Test the condition: `j <= 2`? Yes.	0	1	`–2 –1 0`
Set `k` to the value that `j` has.	0	0	`–2 –1 0`
Test the condition: `k` < `j + 3`? Yes.	0	0	`–2 –1 0`
Output `k`.	0	0	`–2 –1 0 0`

At this point, stop because (E) is the only choice that starts `–2 –1 0 0`. However, repeating this process will result in the correct output of `–2 –1 0 0 1 2 2 3 4`. The correct answer is (E).

11. **A**

To solve this problem, first note that `count % 3` is equal to the remainder when `count` is divided by 3. Execute the method call `mystery(5, "X")`, taking `count = 5` and `s = "X"` as parameters. Because it is not the case that `5 <= 0`, do not execute the first `if` statement.

Because 5 % 3 = 2, calling `mystery(5, "X")` will execute the `else` statement, printing `X`. Then, it will call `mystery(4, "X")`, and return. Because 4 % 3 = 1, calling `mystery(4, "X")` will execute the `else if` statement, printing `X–X`, call `mystery(3, "X")`, and return. At this point, (C) and (E) can be eliminated. Because 3 % 3 = 0, calling `mystery(3, "X")` will execute the `if` statement, printing `X–X`, call `mystery(2, "X")`, and return. Note that you can stop at this point because (B) and (D) can be eliminated. If you're interested, here is the remainder of the execution.

Because $2 \% 3 = 2$, calling `mystery(2, “X”)` will execute the `else` statement, printing `X`, call `mystery(1, “X”)`, and return. Because $1 \% 3 = 1$, calling `mystery(1, “X”)` will execute the `else if` statement, printing `X–X`, call `mystery(0, “X”)`, and return. Finally, calling `mystery(0, “X”)` will simply return because count is less than or equal to zero. Putting it all together, `mystery(5, “X”)` prints

```
XX–XX–XXX–X
```

The correct answer is (A).

12. **B**

The only information that you need to solve this problem is the first sentence in the description of the constructor. Class `DiningRoomSet` has a constructor, which is passed a `Table` object and an `ArrayList` of `Chair` objects. Because you are writing a constructor, you can immediately eliminate (A) and (C), which are `void` methods. Constructors never return anything, so there is never a need to specify a `void` return. Choice (E) is incorrect because the constructor is passed a `Table` object and a `Chair` object, not a `Table` object and an `ArrayList` of `Chair` objects. Choice (D) is incorrect because the second parameter has two types associated with it. It should have only one type: `ArrayList`. Choice (B) is correct.

13. **D**

The best way to solve this problem is to eliminate choices. Choices (A) and (B) are incorrect because the class description states that the `getPrice` method of the `DiningRoomSet` class does not take any parameters. Choice (C) can be eliminated because the private data field `myChairs` is not a `Chair`; it is an `ArrayList`.

Therefore, it does not have a `getPrice` method. This leaves (D) and (E). You need to know that `ArrayLists` are accessed using the `get` method while arrays are accessed using the `[]` notation. `MyChairs` is an `ArrayList`, so the correct answer is (D).

14. **C**

To solve this type of problem, use information about the output and the loops to eliminate incorrect choices. Then, if necessary, work through the code for any remaining choices to determine the correct answer. There are six lines of output. By examining the outer loop of each of the choices, you can eliminate (B) because it will traverse the loop seven times, and during each traversal at least one number will be printed. You can also eliminate (A) because the outer loop will never be executed. The initial value of j is 6, but the condition `j < 0` causes the loop to terminate immediately. The remaining choices have the same outer loop, so turn your attention to the inner loop. Eliminate (D) because the first time through the loop, a 7 will be printed—clearly not the correct output. Finally, eliminate (E) because the condition of the inner loop, `k >= 0`, will cause a 0 to be printed at the end of each line. This leaves (C), which is indeed the correct answer.

15. **B**

This problem tests your knowledge of the methods of the `ArrayList` class. The following table shows the contents of `list` after each line of code.

Code	**Contents of list**	**Explanation**

`list = new ArrayList();`	`[]`	A newly created `ArrayList` is empty
`list.add(new Integer(7));`	`[7]`	Adds 7 to the end of `list`
`list.add(new Integer(6));`	`[7, 6]`	Adds 6 to the end of `list`
`list.add(1, new Integer(5));`	`[7, 5, 6]`	Inserts 5 into `list` as position 1, shifting elements to the right as necessary
`list.add(1, new Integer(4));`	`[7, 4, 5, 6]`	Inserts 4 into `list` as position 1, shifting elements to the right as necessary
`list.add(new Integer(3));`	`[7, 4, 5, 6, 3]`	Adds 3 to the end of `list`
`list.set(2, new Integer(2));`	`[7, 4, 2, 6, 3]`	Replaces the number at position 2 in `list` with 2
`list.add(1, new Integer(1));`	`[7, 1, 4, 2, 6, 3]`	Inserts 1 into `list` at position

		1, shifting elements to the right as necessary

The next command is to print `list`. Therefore, the correct answer is (B).

16. **E**

Although the `Dog` class extends `Animal`, Java does not require `Dog` to have any specific methods. New methods will be unique to `Dog` objects (and those of its subclasses, when applicable) and methods inherited from `Animal` can be invoked as normal.

The correct answer is (E).

17. **A**

This question is testing your understanding of static and dynamic types.

```
Fish Bob = new Shark();
```

This line creates the `Bob` variable with the static type of `Fish` and the dynamic type of `Shark`.

```
System.out.println(Bob.endoskeleton);
```

This line prints the `endoskeleton` field of the variable `Bob`. When looking up a field, the static type is used, so `“bone”` is printed.

```
Bob.action();
```

This line executes the method `action()`. When looking up a method, the dynamic type is used, so "`chomp chomp`" is printed. The correct answer is (A).

18. **D**

The insertion sort algorithm creates a sorted array by sorting elements one at a time.

The `for` loop of the code shows that the elements are being sorted from left to right. To determine the position of the new element to be sorted, the value of the new element must be compared with the values of the sorted elements from right to left. This requires `index–` in the `while` loop, not `index++`. Choices (A) and (C) are wrong.

In order to place the new element to be sorted in its correct position, any sorted elements larger than the new element must be shifted to the right. This requires `sort[index] = sort[index -1]`. Choices (B) and (E) are wrong. The correct answer is (D).

19. **C**

In order for the array to be sorted in descending order, you will need to make a change. Plug each answer choice into the array to see which is correct. In (A) and (B), as the index will never be less than 0, the contents of the `while` loop will never be executed. Choice (C) is the correct answer, as it changes the condition of finding the position of the new element from being less than the compared element to greater. In (D), altering the `for` loop this way would lead to the elements being sorted from right to left but still in ascending order. In (E), the index is initialized at 1 and decremented while `index` > 0, so it will execute the `for` loop only once. The correct answer is (C).

20. **C**

The rate at which insertion sort runs depends on the number of comparisons that are made. The number of comparisons is minimized with an array with elements that are already sorted in ascending order and maximized with an array with elements that are sorted in descending order. The array in (C) is sorted in reverse order and will require the most comparisons to sort. The correct answer is (C).

21. **C**

All three responses look very similar, so look carefully at the difference. Statement I is missing the keyword `new,` which is needed to create a new object. Eliminate any choice that includes Statement I: (A) and (D). Statements II and III differ in that Statement II uses `f.numerator` and `f.denominator` while Statement III uses `f.getNumerator()` and `f.getDenominator().` Since the integers `numerator` and `denominator` are private, while the methods `getNumerator()` and `getDenominator()` are public, the methods must be called by another object. Therefore, Statement II is not valid. Eliminate any choice that includes it: (B) and (E). The correct answer is (C).

22. **C**

Go through the choices one at a time. Choice (A) is incorrect because the multiplication operator, $*$, is not defined to work on `Fraction` objects. Eliminate (A). Choice (B) is incorrect because the multiply method takes only one parameter and it is not correctly invoked by a `Fraction` object. Eliminate (B). Choice (C) correctly calls the `multiply()` method as through a fraction object, taking the other fraction as a parameter. Keep (C). Choice (D) attempts to create a new `Fraction` object but incorrectly constructs it by passing two `Fraction` objects rather than two integers. Eliminate (D). Finally, while (E) calculates the value of the result of the multiplication, the "`/`" operator assigns integer types, which cannot be applied to `answer`, which is an object of type `Fraction`. Eliminate (E). The correct answer is (C).

23. **D**

Go through each statement one at a time. Constructor I is legal. This default constructor of the `ReducedFraction` class will automatically call the default constructor of the `Fraction` class and the private data of both classes will be set appropriately. Eliminate any choice that does not include Constructor I: (B), (C), and (E). Because Constructor II is not included in the remaining choices, do not worry about analyzing it. Constructor III is also legal. The call `super(n, d)` invokes the second constructor of the `Fraction` class, which sets the private data of the `Fraction` class appropriately. The private data of the `ReducedFraction` class is then set explicitly in the `ReducedFraction` constructor. Note that if the call `super(n, d)` were not present, Constructor III would still be legal. However, it would create a logical error in the code, as the default constructor of the `Fraction` class would be invoked, and the private data of the `Fraction` class would not be set appropriately. Eliminate (A), which does not include Constructor III. Only one choice remains, so there is no need to continue. However, to see why Constructor II is illegal, remember that derived classes may not access the private data of their super classes. In other words, the constructor for a `ReducedFraction` may not directly access `numerator` and `denominator` in the `Fraction` class. The correct answer is (D).

24. **D**

Go through each statement one at a time. Statement I is incorrect. “==” checks object references as opposed to their contents. This is an important distinction as two different objects may hold the same data. Eliminate (A), (C), and (E), which contain Statement I. Both of the remaining choices contain Statement II, so don’t worry about analyzing it. Statement III is correct. The `compareTo()` method of the `String` class returns 0 if the two `String` objects hold the same strings. Eliminate (B), which does not include Statement III. Only one choice remains, so there is no need to continue. However, you should see why Statement II is also correct. The `equals` method of the `String` class returns true if the two `String` objects hold the same strings. The correct answer is (D).

25. **A**

The values of `s` and `t` are not changed in `mystery().` Even though `s` and `t` are both parameters of the method, only the instance variables `a` and `b` are changed. Thus, the original Strings `s` and `t` are unaffected by any action in `mystery().` The correct answer is (A).

26. **E**

Though these two classes are related through an inheritance relationship, there is no rule in Java that requires this structure to share methods and/or data. Therefore, there are no requirements on either class. The correct answer is (E).

27. **B**

An example will help clarify this question.

Consider the following `for` loop:

```
for (int k = 0; k < 3; k++)
{
     Systemout.println(k);
}
```

This prints out the integers 0 to 2, one number per line. Matching `int k = 0` to <1>; `k < 3` to <2>; `k++` to <3> and `System.out.println(k);` to <4>, go through each choice one at a time and determine whether each has the same functionality.

Choice (A) initializes `k` and assigns it the value 0. Then it tests to determine whether `k` < 3. This is true, so execute the `while` loop. The next command is `k++`, so increase `k` by 1 to get `k` = 0 + 1 = 1. Now execute `System.out.println(k)` to print `1`. However, the first number printed in the original was `0`, so this is incorrect. Eliminate (A).

Choice (B) initializes `k` and assigns it the value 0. Then it tests to determine whether `k` < 3. This is true, so execute the `while` loop. Now execute `System.out.println(k)` to print `0`. The next command is `k++`, so increase `k` by 1 to get `k` = 0 + 1 = 1. Go back to the top of the `while` loop. Since `k` = 1, it is still the case that `k` < 3, so execute the `while` loop. Now execute `System.out.println(k)` to print `1`. The next command is `k++`, so increase `k` by 1 to get `k` = 1 + 1 = 2. Go back to the top of the `while` loop. Since `k` = 2, it is still the case that `k` < 3, so execute the `while` loop. Now execute `System.out.println(k)` to print `2`. The next command is `k++`, so increase `k` by 1 to get `k` = 2 + 1 = 3. Go back to the top of the `while` loop. Since `k` = 3, it is no longer the case that `k` < 3, so stop executing the `while` loop. The result of the program is printing the integers 0 to 2, one number per line, so keep (B).

Choice (C) initializes `k` and assigns it the value 0. Then it tests the condition `! (k < 3)`. Since it is the case that `k` < 3, the statement `(k < 3)` has the boolean value `true`, making the boolean value of `! (k < 3) false`. Thus the `while` loop is not executed. Since the `while` loop is not executed, nothing is printed. Eliminate (C).

Choice (D) has the same initialization statement and `while` loop condition as (C), so nothing is printed by this choice either. Eliminate (D).

Choice (E) initializes `k` and assigns it the value 0. The next command is `k++`, so increase `k` by 1 to get `k` $= 0 + 1 = 1$. Then it tests to determine whether `k` < 3. This is true, so execute the `while` loop. Execute `System.out.println(k)` to print `1`. However, the first number printed in the original was 0, so this is incorrect. Eliminate (E).

The correct answer is (B).

28. **A**

This question tests your knowledge of operator precedence. In Java, multiplication, division, and modulus are performed before addition and subtraction. If more than one operator in an expression has the same precedence, the operations are performed left-to-right. Parenthesizing the expression one step at a time

```
     a / b + c − d % e ∗ f
→    (a / b) + c − d % e ∗ f
→    (a / b) + c − (d % e) ∗ f
→    (a / b) + c − ((d % e) ∗ f)
→    ((a / b) + c) − ((d % e) ∗ f)
```

The correct answer is (E).

29. **B**

Come up with a sample array of 10 strings with length 5. Let `String[] x` be `{“gator”, “teeth”, “ducky”, “quack”, “doggy”, “woofs”, “kitty”, “meows”, “bears”, “growl”}.` The code must print the first letter of all 10 strings, followed by the second letter of all 10 strings, and so on. Therefore, it must print

```
gtdqdwkmbgaeuuooieer…
```

Go through each segment one at a time.

Segment I initializes `i` and `j` with no values. The outer `for` loop sets `i = 0`, and the inner `for` loop sets `j = 0`. The instruction of the inner `for` loop is `System.out.print(x[i].substring(j, j + 1)`. The `substring(a, b)` method of the `String` class returns a string made up of all the characters starting with the index of `a` through the index of `b – 1`. Therefore, `x[i].substring(j, j + 1)` returns the characters of `x[i]` from index `j` through index `(j + 1) – 1`. Since `(j + 1) – 1 = j`, it returns all the characters from indexes `j` through `j`—a single character string made up of the character at index `j`. Therefore, if `i` = 0 and `j` = 0, `System.out.print(x[i].substring(j, j + 1)` returns the character at index 0 of the string at index 0. This is the `“g”` from `“gator”`. This is what should be printed, so continue. The inner `for` loop increments `j` to get `j` = 1. Since `j` < 5, the loop is executed again, printing the character at index 1 of the string at index 0. This is the character `“a”` from `“gator”`. However, the character `“t”` from `“teeth”` should be the next character printed, so Segment I does not execute as intended. There is no need to continue with I, but note that it will eventually print all of the characters of the first string followed by all of the characters of the second string, and so on. Eliminate (A), (C), and (E), which include I.

Both remaining choices include II, so don't worry about checking this one. Instead, analyze III.

Segment III initializes `i` and `j` with no values. The outer `for` loop sets `i` = 0, and the inner `for` loop sets `j` = 0. The instruction of the inner `for` loop is `System.out.print(x[i].substring(j, j + 1)`. As discussed above, when `i` = 0 and `j` = 0, this command returns the character at index 0 of the string at index 0, which is the "`g`" from "`gator`". This is what should be printed, so continue. The inner `for` loop increments `j` to get `j` = 1. Since `j` < 5, the loop is executed again, printing the character at index 1 of the string at index 0. This is the character "`a`" from "`gator`". Again, the character "`t`" from "`teeth`" should be the next character printed, so Segment III does not execute as intended. There is no need to continue with Segment III, but note that it will attempt to print the first 10 characters of the first 5 strings. Since each String contains only 5 characters, an `IndexOutOfBoundsException` will be thrown when `j` = 5 and the program attempts to print the character at index 5. Eliminate (D), which includes Segment III.

Only one answer remains, so there is no need to continue. However, to see why Segment II is correct, note that it is similar to Segment I, but it reverses the roles of `i` and `j`. The command `System.out.print(x[j].substring(i, i + 1))` prints the character at index `j` of the string at index `i`. The inner `for` loop increments the index of the `String` array before the `outer` loop increments the index of the character in each `String`. Therefore, Segment II correctly prints out the first character of all 10 strings followed by the second character of all 10 strings, and so on. The correct answer is (B).

30. **A**

Begin with (D) and (E), which discuss whether an error would occur. Because `c` is initialized with the command in the third line, `double c;`, eliminate (D). It is perfectly legal to assign a value of type `int` to a variable of type `double` in an expression, so eliminate (E). However, certain rules apply when evaluating the expression. In the expression `c = a / b;` `a` and `b` are both integers. Therefore, the `/` operator represents integer division. The result of the division is truncated by discarding the fractional component. In this example, 7 / 4 has the value 1—the result of truncating 1.75. When assigning an integer to a variable of type `double,` the integer value is converted to its equivalent double value. Therefore, the correct answer is (A).

31. **B**

For the code to print the word `Yes` two conditions must be true. The remainder must be zero when `x` is divided by 2. In other words, `x` must be divisible by 2: that is, even. The value after truncating the result when `x` is divided by 3 must be 1. In other words, $1 \leq x / 3 < 2$. The second condition is more narrow than the first. The only integers that fulfill the second condition are 3, 4, and 5. Of those, only 4 is even and therefore also fulfills the first condition. Therefore, the correct answer is (B).

32. **B**

Go through each statement one at a time. Segment I is incorrect because all parameters in Java are passed by value. After a method returns to its caller, the value of the caller's parameters is not modified. The strings will not be swapped. Eliminate (A), (C), and (E), which include Segment I. Since both remaining choices include Segment III, don't worry about it. Segment II is incorrect because `myName` is a private data field of the `SomeClass` class and may not be accessed by the `swap` method. Note that the question specifically states that the `swap` method is not a method of the `SomeClass` class. Eliminate (D), which includes Segment II. Only one choice remains, so there is no need to continue. To see that Segment III correctly swaps the names of the objects, note that it calls the public methods `setName()` and `getName()` rather than the private `String myName`. Because the references of the instance object variables are the same as the references of the class object variables, modifying the data of the instance variables also changes the data of the class variable. The correct answer is (B).

33. **E**

The way to approach this type of design problem is to look for HAS-A and IS-A relationships among the distinct pieces of data. A book HAS-A title and author. The title and author should be data fields of the `Book` class, either as `Strings` or as their own unrelated classes. This information is not enough to answer the question though. Looking for the IS-A relationships, a mystery novel IS-A work of fiction, which IS-A book. Therefore, it makes good design sense for these three items to be separate classes. Specifically, `Mystery` should be a subclass of `FictionWork`, which should be a subclass of `Book`. Similarly, `RomanceNovel` and `ScienceFiction` should be subclasses of `FictionWork` and `Biography`, `Cookbook`, and `SelfHelpBook` should be subclasses of `NonFictionWork`, which should be a subclass of `Book`. Only (E) meets all of these design criteria.

34. **C**

In order to solve this recursive problem, work backward from the base case to the known value of `mystery(4)`.

Let y represent the <missing value>. Note that `mystery(1)` = y. Now calculate `mystery(2), mystery(3),` and `mystery(4)` in terms of y.

```
mystery(2) = 2 * mystery(1) + 2
     = 2 * y + 2
mystery(3) = 2 * mystery(2) + 3
     = 2 * (2 * y + 2) + 3
     = 4 * y + 4 + 3
     = 4 * y + 7
mystery(4) = 2 * mystery(3) + 4
     = 2 * (4 * y + 7) + 4
     = 8 * y + 14 + 4
     = 8 * y + 18
```

Because `mystery(4)` also equals 34, set 8 * y + 18 = 34 and solve for y.

```
8 * y + 18 = 34
8 * y = 16
y = 2
```

The correct answer is (C).

35. **E**

The `for` loop terminates when the condition is no longer true. This can happen either because `k` is no longer less than `X.length` or because `X[k]` does not equal `Y[k].` Choice (E) states this formally. Another way to approach the problem is to use DeMorgan’s Law to negate the condition in the `for` loop. Recall that DeMorgan’s Law states that

```
!(p && q) is equivalent to !p || !q
```

Negating the condition in the `for` loop gives

```
!(k < X.length && X[k] == Y[k])
=> !(k < X.length) || !(X[k] == Y[k])
=> k >= X.length || X[k] != Y[k]
```

This method also gives (E) as the correct answer.

36. **E**

Choices (A), (B), (C), and (D) are examples of an `ArithmeticException`, a `nullPointerException`, an `ArrayIndexOutOfBoundsException`, and an `IndexOutOfBoundsException`, respectively. Be careful! While (E) may appear to be an example of an `IllegalArgumentException`, it is actually an example of an error that is caught at compile time rather than at runtime. An `IllegalArgumentException` occurs when a method is called with an argument that is either illegal or inappropriate—for instance, passing –1 to a method that expects to be passed only positive integers. Therefore, the correct answer is (E).

37. **E**

First note that `a` and `b` are of type `Double`, not of type `double`. This distinction is important. The type `double` is a primitive type; the type `Double` is a subclass of the `Object` class that implements the `Comparable` interface. A `Double` is an object wrapper for a `double` value. Go through each choice one at a time. Choice (A) is incorrect. It compares `a` and `b` directly rather than the `double` values inside the objects. Even if `a` and `b` held the same value, they might be different objects. Eliminate (A). Choice (B) is incorrect. The `Double` class does not have a `notEquals` method.

Eliminate (B). Choice (C) is incorrect, because `a.doubleValue()` returns a double. Since `double` is a primitive type, it does not have any methods. Had this choice been `!(a.equals(b))`, it would have been correct.

The expression `a.compareTo(b)` returns a value less than zero if `a.doubleValue()` is less than `b.doubleValue()`, a value equal to zero if `a.doubleValue()` is equal to `b.doubleValue()`, and a value greater than zero if `a.doubleValue()` is greater than `b.doubleValue()`. Choice (D) is incorrect because the `compareTo` method returns an int rather than a boolean. Choice (E) is correct. Since `compareTo()` returns `0` if and only if the two objects hold the same values, it will not return `0` if the values are different. The correct answer is (E).

38. **A**

While "encapsulating functionality in a class" sounds like (and is) a good thing, "declaring all data fields to be public" is the exact opposite of good programming practice. Data fields to a class should be declared to be private in order to hide the underlying representation of an object. This, in turn, helps increase system reliability. Choices (B), (C), (D), and (E) all describe effective ways to ensure reliability in a program. The correct answer is (A).

39. **E**

Go through each method one at a time. Method 1 examines at most 10 words using a binary search technique on 1,024 pages to find the correct page. Remember, the maximum number of searches using binary search is $\log_2(n)$ where n is the number of entries. It then searches sequentially on the page to find the correct word. If the target word is the last word on the page, this method will examine all 50 words on the page. Therefore, Method 1 will examine at most 60 words. Eliminate (A), (B), and (C), which don't have 60 for Method 1. The two remaining choices both have the same number for Method 2, so worry only about Method 3. Method 3 sequentially searches through all of the words in the dictionary. Because there are 51,200 words in the dictionary and the target word may be the last word in the dictionary, this method may have to examine all 51,200 words in order to find the target word. Eliminate (D), which does not have 51,200 for Method 3. Only one choice remains, so there is no need to continue. However, note that Method 2 first uses a sequential search technique to find the correct page. If the target word is on the last page, this method will examine the first word on all 1,024 pages. Then, as with Method 1, it may examine as many as all 50 words on the page to find the target word. Therefore, Method 2 will examine at most 1,074 words. The correct answer is (E).

40. **C**

Pick a positive value of `j`. The question says that `j` is a positive integer. A recursive method is easiest if it takes fewer recursions to get to the base case, so try `j` = 1. If `j` = 1, then `m` = 2`j` = 2(1) = 2. Determine the return of `mystery(2)`. Since it is not the case that `m` = 0, the `if` condition is false, so execute the `else` statement, which is to return 4 + `mystery(m – 2)`. Since `m` – 2 = 2 – 2 = 0, `mystery(2)` returns 4 + `mystery(0)`. Determine the return of `mystery(0)`. In `mystery(0)`, `m` = 0, so the `if` condition is true, which causes the method to return 0. Therefore, `mystery(0)` returns 0, and `mystery(2)` returns 4 + `mystery(0)` = 4 + 0 = 4. Go through each choice and eliminate any that are not 4. Choice (A) is 0, so eliminate (A); (B) is `m`, which is 2, so eliminate (B). Choice (C) is 2`m`, which is 2(2) = 4, so keep (C). Choice (D) is `j`, which is 1, so eliminate (D). Choice (E) is `2j`, which is 2(1) = 2, so eliminate (E). Only one choice remains. The correct answer is (C).

Section II: Free-Response Questions

1. `DayCare`—Canonical Solution

```
public class DayCare
{
     private ArrayList<Employee> employees;
     private ArrayList<Child> children;
     private int maxRatio;

     public DayCare(int maxRatio)
     {
          employees = new ArrayList<Employee>();
          children = new ArrayList<Child>();
          this.maxRatio = maxRatio;
     }
```

(a)

```
      public boolean findEmployeeForChild(Child c)
     {
     for (Employee e : employees)
     {
          if (e.childrenAssigned() < maxRatio &&
          e.canTeach(c.getAge()))
          {
               e.assignChild(c);
               return true;
          }
     }

     return false;
}
```

This can also be done with a typical `for` loop or a `while` loop and the `.get` method. A loop is necessary to look at each `Employee` in the `ArrayList`. You must then make sure the chosen `Employee` doesn't already have the maximum number of children allowed. You must also send the age of the `Child` using the `getAge` accessor to the `canTeach` method to see whether the chosen `Employee` is eligible to teach the given `Child`.

If an `Employee` is found for the given `Child`, you need to assign the `Child` to the `Employee` using the `assignChild` method and return `true`.

You should not return `false` inside the loop, since it is possible a different `Employee` is eligible to teach the given `Child`.

(b)
```
public boolean runDayCare()
{
     for (Child c : children)
     {
          if (findEmployeeForChild(c) == false)
          return false;
     }

     return true;
}
```

This could also be done with a typical `for` loop or a `while` loop and the `.get` method. A loop is needed to look at each `Child` in the `ArrayList`. You must then call the `findEmployeeForChild` method from part **(a)** to see whether there is an `Employee` eligible to teach the current `Child`. If not, you need to return `false`. You shouldn't return `true` inside the loop, since it is possible there is a later `Child` who can't be taught by any of the `Employees` in the `ArrayList`.

(c)
```
public boolean addChild(Child c)
{
   if (findEmployeeForChild(c) == true)
```

```
            {
                children.add(c);
                return true;
            }

            return false;
        }
    }
```

The solution must call the `findEmployeeForChild` method from part **(a)** to see whether there is an `Employee` eligible to teach the given `Child`. If there is, you add the `Child` to the `ArrayList` using the `add` method and return `true`. If there isn't, you return `false`.

`DayCare` Rubric

Part (a)

(+1) `for` loop correctly iterates through all elements of the `employees` list

(+1) `if` statement correctly determines whether the number of children assigned to the current `employee` is less than the maximum allowed

(+1) `if` statement correctly determines whether current employee can teach the given child based on age

(+1) The child is assigned to the employee if both conditions are met

(+1) `true` is returned if the child is assigned to an employee; `false` is returned otherwise

Part (b)

(+1) `for` loop correctly iterates through all elements of the `children` list

(+1) The `findEmployeeForChild` method is called correctly

Part (c)

(+1) The `findEmployeeForChild` method is called correctly
(+1) Children are added to the `children` list correctly if an employee is found

2. `Person`—Canonical Solution

```
public class Player extends Person
{
     private String position;

     public Player(String name, int age, String pos)
     {
          super(name, age);
          position = pos;
     }

     public void changePosition(String p)
     {
          position = p;
     }
}
```

The class header must look exactly the same as the header above. The `public class Player` part would be necessary for any class you are writing called `Player`. The `extends Person` part is necessary because a `Player` is a `Person`.

The `position` variable **must** be declared as `private` and it must be a string.

The constructor (`public Player`) must take three parameters in the order shown above, since the example shows the name, age, and position in that order. They can be called whatever you want, however. The `name` and `age` variables must be sent to the `Person` class using the `super( )` call and they must be in the given order. The `position` variable should be set after the `super( )` call.

The `changePosition` method should be `void` and should take a string parameter. The only thing it needs to do is set the class-level `position` variable.

`Person` Rubric

(+1) `public class Player`
(+1) `extends Person`
(+1) A string variable is declared as `private`
(+1) The constructor header is correct (`public Player`).
(+1) The constructor takes a string parameter, an integer parameter, and a string parameter, in that order
(+1) The constructor call uses `super` to initialize the name and age
(+1) The constructor initializes the class-level string variable
(+1) The header for `changePosition` is correct
(+1) The `changePosition` method correctly modifies the class-level string variable

3. `fullName`—Canonical Solution

(a)
```
public static String getFirstName(String name)
{
     int space = name.indexOf(“ ”);
     String first = name.substring(0, space);
     return first;
}
```

You need to use the `indexOf` method to find the location of the space. Once you know where the space is located, you can use the `substring` method to extract from the beginning of the name (index `0`) up to the space. Since the second parameter of the `substring` method is excluded, the space will not be included when the first name is returned.

(b)
```
public static String getLastName(String name)
{
     int space = name.indexOf(" ");
     String last = name.substring(space + 1);
     return last;
}
```

The solution to part **(b)** is similar to that of part **(a)**. You still need to find the location of the space, but the substring starts at the location of the space plus `1`, which will be the first letter of the last name. You could add a second parameter of `name.length()`, but it isn't required.

(c)
```
public static int countVowels(String name)
{
     int count = 0;
     for (int i = 0; i < name.length(); i++)
     {
          String letter = name.substring(i, i + 1);
          if (letter.equals("a") || letter.equals("e") ||
          letter.equals("i") || letter. equals("o") ||
          letter.equals("u"))
               count++;
     }
     return count;
}
```

You need to create a loop to go through the entire name. You could use a `for-each` loop to extract characters, but characters are not part of the AP subset. If you use them, make sure you use them correctly. With a traditional `for` loop, you need to extract each letter using the `substring` method and see whether it equals one of the vowels. A count variable is increased by `1` each time a vowel is found.

`fullName` Rubric
Part (a)

(+1) The `indexOf` method is used correctly to find the first space
(+1) The `substring` method is used correctly to extract the first name
(+1) The first name is returned correctly

Part (b)

(+1) The `indexOf` method is used correctly to find the first space
(+1) The `substring` method is used correctly to extract the last name
(+1) The last name is returned correctly

Part (c)

(+1) A loop is used to look at each letter in the name
(+1) An `if` statement is used to determine whether the current letter is a vowel
(+1) The correct vowel count is returned

4. `ParkingLot`—Canonical Solution

(a)
```
public int openSpaces()
  {
      int taken = 0;
      for (int r = 0; r < lot.length; r++)
      {
            for (int c = 0; c < lot[r].length; c++)
            {
                  if (lot[r][c] != null)
                        taken++;
            }
```

```
        }
        return (lot.length * lot[0].length) - taken;
    }
```

You need to use nested `for` loops to iterate through the `lot` 2D array. You could also use `lot[0].length` in the second `for` loop that iterates through the columns instead of `lot[r].length`. If you find a spot that is not equal to `null` (meaning there is already a car parked there), then you should increase a counter variable by `1`. That variable should be subtracted from the size of the 2D array to get the final result.

(b)
```
public boolean parkCar(Car newCar)
{
     if (openSpaces() > 0)
     {
          for (int r = 0; r < lot.length; r++)
          {
               for (int c = 0; c < lot[r].length; c++)
               {
                 if (lot[r][c] == null)
                 {
                      lot[r][c] = newCar;
                      return true;
                 }
               }
          }
     }

     return false;
}
```

You need to use nested `for` loops to iterate through the `lot` 2D array. You could also use `lot[0].length` in the second `for` loop that iterates through the columns instead of `lot[r].length`. If you find a spot that is `null`, you need to set that spot to `newCar` and return `true`, in that order. There should be a `return false` at the end of the method or in an `else` statement. On the AP exam, if a question tells you to use a method (such as `openSpaces`) and you don't use it, you can lose points.

`ParkingLot` Rubric

Part (a)

(+1) A variable is declared to keep track of the taken parking spaces
(+1) Nested `for` loops are used correctly to iterate through the `lot` 2D array
(+1) An `if` statement checks whether the current spot in the array is not `null`
(+1) The correct number of open spaces is returned

Part (b)

(+1) The `openSpaces` method is called correctly to determine whether there are spaces available
(+1) Nested `for` loops are used correctly to iterate through the `lot` 2D array
(+1) An `if` statement checks whether the current spot in the array is `null`
(+1) The `newCar` is assigned correctly to a `null` element in the 2D array
(+1) The method returns `true` if a spot was found, and `false` otherwise

HOW TO SCORE PRACTICE TEST 3

Section I: Multiple-Choice

______ Number Correct (out of 40)	× 1.875 =	______ Weighted Section I Score (Do not round)	

Section II: Free-Response

Question 1:	______ (out of 9)	× 2.0833 =	______ (Do not round)
Question 2:	______ (out of 9)	× 2.0833 =	______ (Do not round)
Question 3:	______ (out of 9)	× 2.0833 =	______ (Do not round)
Question 4:	______ (out of 9)	× 2.0833 =	______ (Do not round)

Sum = ______ Weighted Section II Score (Do not round)

AP Score Conversion Chart Computer Science A

Composite Score Range	AP Score
107–150	5
90–106	4
73–89	3
56–72	2
0–55	1

Composite Score

______ Weighted Section I Score + ______ Weighted Section II Score = ______ Composite Score (Round to nearest whole number)