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.

Sample Question

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

Sample Answer

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 >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 element from 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 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. Evaluate the following expression: `4 + 6 % 12 / 4`

(A) 1
(B) 2
(C) 4

(D) `4.5`
(E) `5`

2. Which of the following expressions does NOT evaluate to `0.2`?

(A) `(1.0 * 2) / (1.0 * 10)`
(B) `2.0 / 10`
(C) `(double) 2 / 10`
(D) `(double)(2 / 10)`
(E) `Math.sqrt(4) / Math.sqrt(100)`

3. Choose the code used to print the following:

```
"Friends"
```

(A) `System.out.print(""Friends"");`
(B) `System.out.print("//"Friends//"");`
(C) `System.out.print("/"Friends/"");`
(D) `System.out.print("\"Friends\"");`
(E) `System.out.print("\\"Friends \\"");`

4. Determine the output of the following code.

```
String animal1 = "elephant";
String animal2 = "lion";
swap(animal1, animal2);
animal1.toUpperCase();
animal2.toLowerCase();

System.out.println(animal1 + "   " + animal2);

public static void swap(String a1, String a2) {
     String hold = a1;
     a1 = a2;
     a2 = hold;
}
```

(A) `elephant  lion`
(B) `ELEPHANT  lion`
(C) `lion  elephant`
(D) `LION  elephant`
(E) `LION  ELEPHANT`

Questions 5–6 refer to the `Constellation class` below.

```
public class Constellation
     private String name;
     private String month;
     private int northernLatitude;
     private int southernLatitude;

     Constellation(String n, String m)
     {
          name = n;
          month = m;
          northernLatitude = 0;
          southernLatitude = 0;
     }

     Constellation(String n, String m, int nLat, int sLat)
     {
          name = n;
          month = m;
          northernLatitude = nLat;
          southernLatitude = sLat;
     }
     public void chgMonth(String m)
     {
          String month = m;
     }
```

5. Using the `Constellation class`, which of the following will cause a compiler error?

(A)
```
Constellation c1 = new Constellation("Hercules",
    "July");
```
(B)
```
Constellation c2 = new Constellation("Pisces", "Nov",
    90, 65);
```
(C)
```
Constellation c3 = new Constellation("Aquarius", "Oct",
    65.0, 90.0);
```
(D)
```
Constellation c4 = new Constellation("Leo", "4", 0, 0);
```
(E)
```
Constellation c5 = new Constellation("Phoenix", "Nov",
    32, 90);
```

6. A programmer has attempted to add three mutator methods to the `Constellation class`.

I.
```
public void chgLatitude(String direction, int latitude)
{
   if (direction.toUpperCase().equals("N"))
        northernLatitude = latitude;
   else if (direction.toUpperCase().equals("S"))
        southernLatitude = latitude;
}
```

II.
```
public void chgLatitude(int nLatitude, int sLatitude)
{
   northernLatitude = nLatitude;
   southernLatitude = sLatitude;
}
```

III.
```
public void chgLatitude(double nLatitude, double
  sLatitude)
{
   northernLatitude = (int) nLatitude;
   southernLatitude = (int) sLatitude;
}
```

Which of the three will compile without a compiler error?

(A) I only

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

7. Determine the values of `x` and `y` after the following code runs.

```
int x = 10;
int y = 5;

if (x == 10)
{
     if (y <= 5)
          y++;
     else if (y < 4)
               x = 3;
          else
               y += 6;
}
if (y > 5)
{
     if (x != 10)
     {
          x = 0;
          y = 0;
     }
     else
          x = -5;
}
```

(A) `x = 0, y = 0`
(B) `x = –5, y = 6`
(C) `x = 10, y = 5`
(D) `x = 3, y = 5`
(E) None of the above

8. A programmer intended to write code to print three words in ascending lexicographical order. Follow the code and determine the printed output.

```
1 String word1 = “frog”;
2 String word2 = “dog”;
3 String word3 = “cat”;
4
5 if (word1.compareTo(word2) < 0)
6    if (word2.compareTo(word3) < 0)
7      System.out.println(word1 + “ ” + word2 + “ ” + word3);
8    else
9      System.out.println(word1 + “ ” + word3 + “ ” + word2);
10 else
11    if (word1.compareTo(word2) > 0)
12      if (word2.compareTo(word3) < 0)
13        System.out.println(word1 + “ ” + word2 + “ ” +
  word3);
14      else
15        System.out.println(word1 + “ ” + word3 + “ ” +
  word2);
16    else
17      if (word2.equals(word3))
18        System.out.println(“all the words are the same”);
19      else
20        System.out.println(“word1 and word2 are duplicates”);
```

(A) `frog  cat  dog`
(B) `cat  dog  frog`
(C) `dog  frog  cat`
(D) `frog  dog  cat`
(E) `dog  cat  frog`

9. Using the following variable declarations, determine which of the following would evaluate to `true`.

```
int temp = 90;
boolean cloudy = false;
```

I. `if (temp >= 90 && !cloudy)`

II. `if (!(temp > 90 || cloudy))`

III. `if (!(temp > 90 && !cloudy))`

(A) I only
(B) II only
(C) III only
(D) Two of the above will evaluate to `true`.
(E) All the above will evaluate to `true`.

10. Consider the following code:

```
1 String dog1 = new String(“Poodle”);
2 String dog2 = new String(“Beagle”);
3 dog1 = dog2;
4 String dog3 = new String(“Beagle”);
5
6 if (dog1 == dog2)
7    System.out.println(“dog1 and dog2 are one and the same
  dog”);
8 else
9    System.out.println(“dog1 and dog2 are not the same dog”);
10
11 if (dog1 == dog3)
12    System.out.println(“dog1 and dog3 are one and the same
  dog”);
13 else
14    System.out.println(“dog1 and dog3 are not the same dog”);
15
16 if (dog1.equals(dog3))
17    System.out.println(“dog1 and dog3 are the same breed”);
18 else
19    System.out.println(“dog1 and dog3 are not the same
  breed”);
```

Which of the following represents the output that will be produced by the code?

(A)
```
dog1 and dog2 are one and the same dog
dog1 and dog3 are one and the same dog
dog1 and dog3 are the same breed
```

(B)
```
dog1 and dog2 are one and the same dog
dog1 and dog3 are one and the same dog
dog1 and dog3 are not the same breed
```

(C)
```
dog1 and dog2 are one and the same dog
dog1 and dog3 are not the same dog
dog1 and dog3 are the same breed
```

(D)
```
dog1 and dog2 are one and the same dog
dog1 and dog3 are not the same dog
dog1 and dog3 are not the same breed
```

(E)
```
dog1 and dog2 are not the same dog
dog1 and dog3 are not the same dog
dog1 and dog3 are the same breed
```

11. Choose the correct option to complete lines 3 and 4 such that `str2` will contain the letters of `str1` in reverse order.

```
1 String str1 = “banana”;
2 String str2 = “”;
3 // missing code
4 // missing code
5 {
6     str2 += str1.substring(i, i + 1);
7     i--;
8 }
```

(A)
```
int i = 0;
while (i < str1.length)
```

(B)
```
int i = str1.length();
while (i >= 0)
```

(C)
```
int i = str1.length() - 1;
while (i >= 0)
```

(D)
```
int i = str1.length();
   while (i > 0)
```
(E)
```
int i = str1.length() - 1;
   while (i > 0)
```

12. Consider the following code excerpt :

```
int n = // some integer greater than zero
int count = 0;
int p = 0;
int q = 0;
for (p = 1; p < n; p++)
   for (q = 1; q <= n; q++)
     count ++;
```

What will be the final value of `count`?

(A) n^n
(B) $n^2 - 1$
(C) $(n - 1)^2$
(D) $n(n - 1)$
(E) n^2

13. Given the following code excerpt, determine the output.

```
1 int x = 0;
2 for (int j = 1; j < 4; j++)
3 {
4    if (x != 0 && j / x > 0)
5      System.out.print(j / x + " ");
6    else
7      System.out.print(j * x + " ");
8 }
```

(A) `0 0 0`
(B) `0 0 0 0`

(C) `1 2 3`

(D) `1 0 2 0 3 0`

(E) `ArithmeticException: Divide by Zero`

14. Consider the following code:

```
1 String space = " ";
2 String symbol = "*";
3 int num = 5;
4 for (int i = 1; i <= num; i++)
5 {
6      System.out.print(symbol);
7 }
8 System.out.print("\n");
9 for (int i = 1; i <= num; i++)
10 {
11   for (int j = num - i; j > 0; j--)
12   {
13        System.out.print(space);
14   }
15   System.out.println(symbol);
16 }
17 for (int i = 1; i <= num; i++)
18 {
19      System.out.print(symbol);
20 }
```

Which of the following represents the output?

(A)
```
*****
****
***
**
*
*****
```

(B)
```
*****
 ****
  ***
   **
    *
*****
```

(C)
```
*****
    *
   *
  *
 *
*****
```

(D)
```
*****
*
 *
  *
   *
*****
```

(E)
```
*****
*
**
***
****
*****
```

15. What will be printed as a result of the following code excerpt?

```
int sum = 0;
for (int i = 1; i < 2; i++)
     for (int j = 1; j <= 3; j++)
          for (int k = 1; k < 4; k++)
               sum += (i * j * k);

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

(A) 18
(B) 36
(C) 45
(D) 60
(E) 108

16. Consider the following code:

```
1 int j = 0;
2 String s = “map”;
3 while (j < s.length())
4 {
5     int k = s.length();
6     while (k > j)
7     {
8        System.out.println(s.substring(j, k));
9        k--;
10    }
11    j++;
12 }
```

Which of the following represents the output?

(A)
```
map
ma
m
ap
```

```
a
```

(B)
```
map
ma
m
ap
a
p
```

(C)
```
map
ap
p
ap
p
p
```

(D)
```
m
ma
map
a
ap
p
```

(E)
```
p
ap
p
map
ma
```

```
m
```

17. A factorial is shown by an exclamation point(!) following a number. The factorial of 5, or $5!$, is calculated by $(5)(4)(3)(2)(1) = 120$.

Assuming `n` is an integer greater than 1, choose the method that will return `n!`

I.	`public static int f(int n) {`
	`  int factorial = 1;`
	`  for (int i = n; i > 0; i--) {`
	`  factorial *= n;`
	`  }`
	`  return factorial;`
	`}`

II.	`public static int f(int n) {`
	`  int factorial = 1;`
	`  int j = 1;`
	`  while (j <= n) {`
	`  factorial *= j;`
	`  j++;`
	`  }`
	`  return factorial;`
	`}`

III.	`public static int f(int n) {`
	`  if (n == 1)`
	`  return n;`
	`  return n * f(n - 1);`
	`}`

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

(E) I, II, and III

Questions 18–20 refer to the code excerpt for the `Tile` class below:

```
1 public class Tile
2 {
3   private int styleNumber;
4   private String color;
5   private double width;
6   private double height;
7   private String material;
8   private double price;

9   Tile(int style, String col)
10   {
11     styleNumber = style;
12     color = col;
13   }
14   Tile(int style, String col, double w, double h, String
  mat, double price)
15   {
16     styleNumber = style;
17     color = col;
18     width = w;
19     height = h;
20     material = mat;
21     price = price;
22   }
23   Tile(int style, String col, String mat, double price)
24     {
25       styleNumber = style;
26       color = col;
27       material = mat;
28       price = price;
29     }
30   public void chgMaterial(String mat)
31   {
32     String material = mat;
33   }
34   public String toString()
```

```
35   {
36     return (styleNumber + “ ” + color + “ ” + width + “ ” +
  height + “ ” +
37          material + “ ” + price);
38   }
39 }
```

18. What is the output after the following client code is executed?

```
Tile t1 = new Tile(785, “grey”, “ceramic”, 6.95);
     t1.chgMaterial(“marble”);
     System.out.print(t1.toString());
```

(A) `Tile@5ccd43c2`
(B) `785 grey 0.0 0.0 marble 0.0`
(C) `785 grey 0.0 0.0 ceramic 0.0`
(D) `785 grey 0.0 0.0 ceramic 6.95`
(E) `785 grey 0.0 0.0 marble 6.95`

19. What is the output after the following client code is executed?

```
Tile t2 = new Tile(101, “blue”);
System.out.print(t2);
```

(A) `Tile@5ccd43c2`
(B) `101 blue 0.0 0.0 null 0.0`
(C) `Type mismatch error`
(D) `NullPointerException`
(E) There will be no output; the program will not compile.

20. The `Tile class` is going to be used for an application built for a small independent tile store. The owner wants the programmer to add a field for the number of unopened boxes of tile he has for each style of tile he has in stock and a method to change the value. What would be the proper declaration for this field?

(A) `public static int inventory;`
(B) `private static double inventory;`
(C) `final int inventory;`
(D) `private int inventory;`
(E) `private int [] inventory;`

21. Given the following code excerpt:

```
9 int[] nums = {11, 22, 33, 44, 55, 66 };
10
11 for (int i = 0; i < nums.length; i++)
12    nums[nums[i] / 11] = nums[i];
```

Determine the final contents of `nums`.

(A) `1, 2, 3, 4, 5, 6`
(B) `11, 11, 33, 33, 55, 55`
(C) `11, 11, 22, 33, 44, 55`
(D) `11, 22, 22, 33, 33, 55`
(E) `11, 22, 33, 44, 55, 66`

22. Given the following code excerpt:

```
13 int[] arr1 = {1, 2, 3, 4, 5, 6 };
14 int[] arr2 = arr1;
15 int last = arr1.length - 1;
16
17 for (int i = 0; i < arr1.length; i++)
18    arr2[i] = arr1[last - i];
19
20 for (int i = 0; i < arr1.length; i++)
21    System.out.print(arr1[i] + "  ");
22
23 System.out.println(" ");
24
25 for (int i = 0; i < arr2.length; i++)
26    System.out.print(arr2[i] + "  ");
```

Determine the statement below that reflects the resulting output.

(A)
```
1  2  3  4  5  6
   1  2  3  4  5  6
```

(B)
```
1  2  3  4  5  6
   6  5  4  4  5  6
```

(C)
```
6  5  4  3  2  1
   6  5  4  4  5  6
```

(D)
```
6  5  4  4  5  6
   1  2  3  4  5  6
```

(E)
```
6  5  4  4  5  6
   6  5  4  4  5  6
```

23. Given the following code excerpt:

```
27 int[] arr3 = {1, 2, 3, 4, 5, 6 };
28
29 for (int element : arr3)
30 {
31    element *= 2;
32    System.out.print(element + "  ");
33 }
34 System.out.println(" ");
35
36 for (int element : arr3)
37    System.out.print(element + "  ");
```

Determine the statement below that reflects the resulting output.

(A)
```
1  2  3  4  5  6
   1  2  3  4  5  6
```

(B)
```
2  4  6  8  10  12
```

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

(C)
```
2  4  6  8  10  12
   2  4  6  8  10  12
```

(D) A compiler error will occur.
(E) A run-time exception will occur.

24. Given an array `numbers` containing a variety of integers and the following code excerpt:

```
38 int holdSmallest = Integer.MAX_VALUE;
39 int holdLargest = 0;
40 int a = 0;
41 int b = 0;
42 for (int i = 0; i < numbers.length; i++)
43 {
44   if (numbers[i] <= holdSmallest)
45   {
46      holdSmallest = numbers[i];
47      a = i;
48   }
49   if (numbers[i] >= holdLargest)
50   {
51      holdLargest = numbers[i];
52      b = i;
53   }
54 }
55 System.out.println(a + " " + b);
```

Determine the statement below that reflects the most successful outcome.

(A) The code will print the smallest and largest values in the `numbers` array.
(B) The code will print the locations of the smallest and largest values in the `numbers` array.

(C) The code will print the locations of the smallest and largest non-negative values in the `numbers` array.

(D) The code will print the locations of the smallest value in the `numbers` array and the largest non-negative value in the `numbers` array.

(E) The code will print the locations of the smallest non-negative value in the `numbers` array and the largest value in the `numbers` array.

25. Choose the missing code below that will accurately find the average of the values in the `sales` array.

```
57 int i = 0;
58 int sum = 0;
59 for (int element : sales)
60
61    //Missing code
62
63
```

(A)
```
{
    sum += element;
}
double avg = (double) sum / sales.length;
```

(B)
```
{
    sum += sales[i];
}
double avg = (double) sum / sales.length;
```

(C)
```
{
    sum += sales;
}
double avg = (double) sum / sales.length;
```

(D)
```
{
    sum += sales[element];
}
```

```
   double avg = (double) sum / sales.length;
(E) {
      sum += element[sales];
   }
   double avg = (double) sum / sales.length;
```

26. A programmer has written two different methods for a client program to swap the elements of one array with those of another array.

```
11   public static void swap1(int[] a1, int[] a2)
12   {
13     for (int i = 0; i < a1.length; i++)
14     {
15       int arrhold = a1[i];
16       a1[i] = a2[i];
17       a2[i] = arrhold;
18     }
19   }
```

```
20
21 public static void swap2(int[] a1, int[] a2) {
22         int [] arrhold = a1;
23         a1 = a2;
24         a2 = arrhold;
25   }
```

Which of the following statements best reflects the outcomes of the two methods?

(A) Both methods will swap the contents of the two arrays correctly in all cases.

(B) `swap1` will swap the contents of the two arrays correctly only if both arrays have the same number of elements, whereas `swap2` will work correctly for all cases.

(C) `swap1` will swap the contents of the two arrays correctly only if both arrays have the same number of elements, whereas

swap2 will never work correctly.

(D) swap1 will swap the contents of the two arrays correctly only if both arrays have the same number of elements or a2 has more elements, whereas swap2 will work correctly for all cases.

(E) Neither method will swap the contents of the two arrays correctly under any conditions.

27. Which code has declared and properly populated the given ArrayList?

I.	`ArrayList <String> alist1 = new ArrayList<String> (); alist1.add("4.5");`
II.	`ArrayList <Integer> alist2 = new ArrayList<Integer>(); alist1.add((int) 4.5);`
III.	`ArrayList <Double> alist3; alist3 = new ArrayList<Double>(); alist3.add(4.5);`

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

28. Given the following code excerpt:

```
ArrayList <Integer> alist1 = new ArrayList<Integer>();
int [] a1 = {2, 4, 6, 7, 8, 10, 11 };
for (int a : a1) {
     alist1.add(a);
}
for (int i = 0; i < alist1.size(); i++) {
     if (alist1.get(i) % 2 == 0){
          alist1.remove(i);
```

```
        }
    }
    System.out.println(alist1);
```

Determine the output.

(A) `[4, 7, 10, 11]`
(B) `[2, 4, 7, 10, 11]`
(C) `[2, 7, 10, 11]`
(D) `[7, 11]`
(E) An `IndexOutOfBoundsException` will occur.

Questions 29–30 refer to the following code excerpt.

```
2 ArrayList <Integer> alist5 = new ArrayList<Integer>();
3 int [] a1 = {21, 6, 2, 8, 1 };
4 for (int a : a1)
5 {
6    alist5.add(a);
7 }
8 for (int k = 0; k < alist5.size() - 1; k++)
9 {
10   for (int i = 0; i < alist5.size() - 2; i++)
11   {
12     if (alist5.get(i) > alist5.get(i + 1))
13     {
14       int hold = alist5.remove(i);
15       alist5.add(i + 1, hold);
16     }
17   }
18 }
19 System.out.println(alist5);
```

29. How many times will line 12 be executed?

(A) 6 times
(B) 12 times

(C) 15 times
(D) 16 times
(E) 20 times

30. What will be the final output after the code executes?

(A) `[21, 8, 6, 2, 1]`
(B) `[6, 21, 2, 8, 1]`
(C) `[6, 2, 8, 21, 1]`
(D) `[2, 6, 8, 21, 1]`
(E) `[1, 2, 6, 8, 21]`

31. Given nums—a rectangular, but not necessarily square, two-dimensional array of integers—consider the following code intended to print the array:

```
4 int [][] arr2d = {{1, 2, 3, 4 }, {5, 6, 7, 8 }};
5 String s = “”;
6 for (int a = 0; a < arr2d[0].length; a++)
7 {
8   for (int b = 0; b < arr2d.length; b++)
9   {
10     s += arr2d [b][a] + “ ”;
11   }
12   s += “\n”;
13 }
14 System.out.print(s);
```

Determine the resulting output.

(A)
```
1  2  3  4
   5  6  7  8
```
(B)
```
1  5  2  6
   3  7  4  8
```
(C)
```
1  2
   3  4
```

```
    5  6
    7  8
(D) 1  5
    2  6
    3  7
    4  8
(E) 1
    2
    3
    4
    5
    6
    7
    8
```

32. Given nums—a rectangular, two-dimensional array of integers—choose the code to print the entire array.

I.	`for (int r = 0; r < nums.length; r++)`
	`{`
	`  for (int c = 0; c < nums[0].length; c++)`
	`  {`
	`  System.out.print(nums[r][c]);`
	`  }`
	`  System.out.print("\n");`
	`}`

II.	`for (int [] row : nums)`
	`{`
	`  for (int col : row)`
	`  {`
	`  System.out.print(col + " ");`
	`  }`
	`  System.out.println("");`
	`}`

III.	`for (int r = 0; r < nums[0].length; r++)` `{`

```
        for (int c = 0; c < nums.length; c++)
        {
            System.out.print(nums[r][c] + " ");
        }
            System.out.print("\n");
    }
```

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

Questions 33–35 refer to the `Percussion class` and `Xylophone class` below.

```
public class Percussion {
    private String name;
    private double weight;
    Percussion() {
    }
    Percussion(String n, double w)
    {
        name = n;
        weight = w;
    }
    public String getName()
    {
        return name;
    }
    public double getWeight()
    {
        return weight;
    }
}
public class Drums extends Percussion
{
}
```

```
public class Xylophone extends Percussion {
     private int numberOfKeys;

     Xylophone(String name, double weight, int
     numberOfKeys){

          <missing code>

     }
     public int getNumKeys()
     {
          return numberOfKeys;
     }
}
```

33. Which of the following is the most appropriate replacement for <missing code> in the `Xylophone` constructor?

(A)
```
this.numberOfKeys = numberOfKeys;
super(name, weight);
```
(B)
```
super(name, weight);
this.numberOfKeys = numberOfKeys;
```
(C)
```
super(name, weight);
numberOfKeys = this.numberOfKeys;
```
(D)
```
this.numberOfKeys = numberOfKeys;
```
(E)
```
numberOfKeys = this.numberOfKeys;
```

34. Assuming the above classes compile correctly, which of the following will NOT compile within a client program?

(A)
```
Xylophone [] xylophones = new Xylophone[5];
```
(B)
```
Percussion [] xylophones = new Xylophone[5];
```
(C)
```
Xylophone x1 = new Xylophone (“xylophone”, 65, 32);
System.out.println(x1.getNumKeys());
```
(D)
```
Xylophone x1 = new Xylophone (“xylophone”, 65, 32);
System.out.println(x1.numberOfKeys);
```

(E) `Drums [] drums;`

35. A client program wishes to compare the two `xylophone` objects as follows:

```
Xylophone x2 = new Xylophone (“xylophone”, 80, 32);
Xylophone x3 = new Xylophone (“xylophone”, 65, 32);
```

The two objects should be considered “equally heavy” if and only if they have the same weight. Which of the following code excerpts accomplishes that task?

(A)
```
if (x2.weight == x3.weight)
    System.out.println(“equally heavy”);
else
    System.out.println(“not equally heavy”);
```

(B)
```
if (x2.weight() == x3.weight())
    System.out.println(“equally heavy”);
else
    System.out.println(“not equally heavy”);
```

(C)
```
if (x2.getWeight() == x3.getWeight())
    System.out.println(“equally heavy”);
else
    System.out.println(“not equally heavy”);
```

(D)
```
if (x2.weight.equals(x3.weight))
    System.out.println(“equally heavy”);
else
    System.out.println(“not equally heavy”);
```

(E) The weights of the objects cannot be compared.

Questions 36–37 refer to the following classes.

```
public class Dog {
    private int height;
    private String size;
    private String color;
```

```
     Dog (int iheight, int iweight, String icolor)
     {
          height = iheight;
          color = icolor;
          if (iweight >= 65)
               size = “large”;
          else
               size = “medium”;
     }
     public int getheight() {return height;}
     public String getSize() {return size;}
     public String getColor() {return color;}
     public String toString() {return “    color is: ” +
     color;}
}

public class SportingDog extends Dog {
     private String purpose;
     SportingDog(int h, int w, String c)
          {
          super(h, w, c);
          purpose = “hunting”;
     }
     public String getPurpose()
     {
          return purpose;
     }
}

public class Retriever extends SportingDog{
     private String type;

     Retriever(String itype, String icolor, int iweight)
     {
          super(24, iweight, icolor);
          type = itype;
     }
          public String toString() {return “    type: ” +
          type + super.toString();}
}
```

36. Which of the following declarations will NOT compile?

(A) `Dog d1 = new SportingDog(30, 74, “Black”);`
(B) `Dog d2 = new Retriever(“Labrador”, “yellow”, 75);`
(C) `SportingDog d3 = new Retriever(“Golden”, “Red”, 70);`
(D) `SportingDog d4 = new Dog(25, 80, “Red”);`
(E) `Retriever d5 = new Retriever(“Golden”, “Blonde”, 60);`

37. What is the output after the execution of the following code in the client program:

```
Dog mason = new Retriever(“Labrador”, “chocolate”, 85);
System.out.println(mason.toString());
```

(A) `type: Labrador`
(B) `type: Labrador, color is: chocolate, purpose: hunting`
(C) `color is: chocolate, type: Labrador`
(D) `type: Labrador, purpose: hunting, color is: chocolate`
(E) `type: Labrador, color is: chocolate`

38. The following `pow` method was written to return `b` raised to the `x`th power where `x > 0`, but it does not work properly. Choose the changes that should be made to the method below so that it works properly.

```
1 public double pow(double b, int x)
2 {
3 if (x == 0)
4   return 1;
5 else
6   return b + pow(b, x – 1);
7 }
```

(A) Change lines 3 and 4 to:

```
3 if (x == 1)
```

```
4   return 1;
```

(B) Change lines 3 and 4 to:

```
3 if (x == 1)
4   return b;
```

(C) Change line 6 to:

```
6   return b * pow(b, x - 1);
```

(D) Both (A) and (C)

(E) Both (B) and (C)

39. What is output given the following code excerpt?

```
System.out.println(f(8765));
public static int f(int n)
{
     if (n == 0)
          return 0;
     else
          return f(n / 10) + n % 10;
}
```

(A) 5678

(B) 8765

(C) 58

(D) 26

(E) A run-time error

40. Choose the best solution to complete the missing code such that the code will implement a binary search to find the variable number in `arr`.

```
int number = <some number in arr>;
System.out.println(search(arr, 0, arr.length - 1,
number));

public int search(int[] a, int first, int last, int
sought) {
```

```
        int mid = (first + last) / 2;

        if (<missing code>) {
            last = mid - 1;
            return search(a, first, last, sought);
        }
        else if (<missing code>)) {
            first = mid + 1;
            return search(a, first, last, sought);
        }

        return mid;
    }
```

(A) `a[mid] > sought, a[mid] < sought`

(B) `a[mid] + 1 > sought, a[mid] < sought`

(C) `a[mid] > sought, a[mid] - 1 < sought`

(D) `a[mid] + 1 > sought, a[mid] - 1 < sought`

(E) `a[mid] = sought, a[mid] = sought`

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.

Section II

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. This question involves the implementation of a simulation of rolling two dice. A client program will specify the number of rolls of the sample size and the number of faces on each of the two dice. A method will return the percentage of times the roll results in a double. *Double* in this case means when two dice match or have the same value (not a data type).

You will write two of the methods in this class.

```
public class DiceSimulation {

    /** Sample size of simulation              */
    private int numSampleSize;

    /** Number of faces on each die            */
    private int numFaces;

    /** Constructs a DiceSimulation where sampleSize is the number of rolls to be simulated and
     *  faces is the number of faces on each die (some dice have more or fewer than 6 faces)
     */
    public DiceSimulation(int numSamples, int faces) {
        numSampleSize = numSamples;
        numFaces = faces;
    }

    /** Returns an integer from 1 to the number of faces to simulate a die roll */
    public int roll() {
                        /* to be implemented in part (a) */
    }

    /** Simulates rolling two dice with the number of faces given, for the number of sample size
     *  rolls. Returns the percentage of matches that were rolled
     *  as an integer (eg. 0.50 would be 50).
     */
    public int runSimulation() {
                        /* to be implemented in part (b) */

    }
}
```

The following table contains sample code and the expected results.

Statements and Expressions	Value Returned / Comment
`DiceSimulation s1 = new DiceSimulation(10, 6)`	(no value returned) A `DiceSimulation s1` is declared and instantiated.
`s1.runSimulation()`	10 rolls are simulated; only the percentage of matches is displayed. See further explanation below.

The 10 rolls might look like this (nothing is printed at this time)

```
Die1:        3          Die2:        4
Die1:        1          Die2:        5
Die1:        2          Die2:        2
Die1:        3          Die2:        4
Die1:        6          Die2:        6
Die1:        3          Die2:        4
Die1:        3          Die2:        3
Die1:        6          Die2:        4
Die1:        3          Die2:        1
Die1:        5          Die2:        5
```

The percentage the method would return is `40.`

(a) Write the `roll` method to simulate the roll of one die.

```
/** Returns an integer from 1 to number of faces to simulate a
die roll */
```

```
public int roll()
```

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.

Class information for this question

```
public class DiceSimulation

private int numSampleSize;
private int numFaces;

public DiceSimulation (int numSamples, int
faces)
public int roll()
public int runSimulation()
```

(**b**) Write the `runSimulation` method.

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.

Class information for this question

```
public class DiceSimulation

private int numSampleSize;
private int numFaces;

public DiceSimulation (int numSamples, int
faces)
public int roll()
public int runSimulation()
```

2. This question involves the implementation of a calorie counter system that is represented by the `CalorieCount` class. A `CalorieCount` object is created with 5 parameters:

   - Daily calories limit—the recommended number of calories per day
   - Daily calories intake—the number of calories a person has eaten in a day
   - Grams of protein per day
   - Grams of carbohydrate per day
   - Grams of fat per day

   The `CalorieCount` class provides a constructor and the following methods:

   - `addMeal`—takes in calories, grams of protein, grams of carbs, and grams of fat from a meal and updates corresponding instance fields

   - `getProteinPercentage`—returns the percent of protein in a given day (4 * grams protein / daily calorie intake)

- `onTrack`—returns `true` if the calorie intake does not exceed the daily calories limit, otherwise returns `false`

The following table contains sample code and the expected results.

Statements and Expressions	Value Returned (blank if no value)	Comment
`CalorieCount sunday = new CalorieCount(1500);`		Creates an instance with a 1500-calorie limit
`sunday.addMeal(716, 38, 38, 45);`		Adds 716 calories, 38 grams of protein, 38 grams of carbs, 45 grams of fat to the appropriate instance fields
`sunday.addMeal(230, 16, 8, 16);`		Adds 230 calories, 16 grams of protein, 8 grams of carbs, 16 grams of fat to the appropriate instance fields
`sunday.addMeal(568, 38, 50, 24);`		Adds 568 calories, 38 grams of protein, 50 grams of carbs, 24 grams of fat to the appropriate instance fields
`onTrack()`	`false`	Returns `true` if calorie intake does not exceed calorie limit

`getProteinPercentage()`	`0.24`	Multiplies grams of protein by 4 and then divides by calorie intake

Write the entire `CalorieCount` class. Your implementation must meet all specifications and conform to all examples.

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. This question involves the implementation of a travel planner system that is represented by the `TravelPlan` and `Tour` classes. A client will create `Tour` objects that will represent tours or activities of interest. Each `Tour` object is made up of an activity date, start time, end time, and name of the activity. The client will also create a `TravelPlan` object comprised of a destination and an `ArrayList` of `Tours`.

   A partial declaration of the `Tour` class is shown below.

```
public class Tour {
     private int actDate;
     private int startTime; // times are represented in
     military format
     private int endTime;   // 1430 for 2:30 pm
     private String activity;

/*  Constructs a Tour
 *   All instance fields are initialized from parameters
 */
```

```
    Tour(int actDate, int startTime, int endTime, String
    activity)
    {
    /*  implementation not shown
    }
    public int getActDate() {return actDate;}
    public int getStartTime() {return startTime;}
    public int getEndTime() {return endTime;}
    public String getActivity() {return activity;}
```

A partial declaration of the `TravelPlan` class is shown below.

```
import java.util.ArrayList;

public class TravelPlan {
    private String destination;
    private ArrayList <Tour> plans;

/*  Constructs a TravelPlan
 *   All instance fields are initialized from parameters
 */
        TravelPlan(String destination)
        {
        /*  to be implemented in part (a)  */
        }

/*  Returns  true  if the timeframe overlaps with another Tour in
plans;
 *  otherwise  false
 */
        public boolean checkForConflicts(Tour t)
        {
        /*  to be implemented in part (b)  */
        }

/* Calls  checkForConflicts,  if  checkForConflicts  returns
false
 *  (the timeframe does not overlap), adds the tour to  plans,
returns  true
```

```
 * otherwise returns false
 * Must call checkForConflicts for full credit
 */
        public boolean addTour(Tour t)
        {
        /* to be implemented in part (c) */
        }
```

The following table contains sample code and the expected results.

Statements and Expressions	**Value Returned (blank if no value)**	**Comment**
`TravelPlan p1 = new TravelPlan("Capetown");`		Creates an instance with a destination "`CapeTown`" and an empty `ArrayList` of type `Tour`
`Tour t1 = new Tour(1312020, 800, 1230, "Bungee jumping");`		Creates a `Tour` instance with date, start time, end time, and activity
`Tour t2 = new Tour(1312020, 900, 1430, "Body surfing");`		Creates a `Tour` instance with date, start time, end time, and activity
`p1.add(t1)`	`true`	Checks for conflicts in `plans`; since there are none, adds the

		`Tour` object, returns `true`
`p1.add(t2)`	`false`	Checks for conflicts in `plans`; since there is a conflict, returns `false`
`Tour t3 = new Tour(2012020, 900, 1200, "Shark cage diving");`		Creates a `Tour` instance with date, start time, end time, and activity
`p1.add(t3)`	`true`	Checks for conflicts in `plans`; since there are none, adds the `Tour` object, returns `true`

(a) Write the `TravelPlan` constructor. The constructor should initialize the destination and the `plans ArrayList`.

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.

Class information for this question

```
public class Tour
private int actDate
```

```
private int startTime
private int endTime
private String activity

Tour(int actDate, int startTime, int endTime,
String activity)
public int getActDate()
public int getStartTime()
public int getEndTime()
public String getActivity()

public class TravelPlan
private String destination;
private ArrayList <Tour> plans;

public TravelPlan(String destination)
public boolean addTour(Tour t)
public boolean checkForConflicts(Tour t)
```

(**b**) Write the `TravelPlan checkForConflicts` method.

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.

Class information for this question

```
public class Tour
private int actDate
private int startTime
private int endTime
private String activity
```

```
Tour(int actDate, int startTime, int endTime,
String activity)
public int getActDate()
public int getStartTime()
public int getEndTime()
public String getActivity()

public class TravelPlan
private String destination;
private ArrayList <Tour> plans;

public TravelPlan(String destination)
public boolean addTour(Tour t)
public boolean checkForConflicts(Tour t)
```

(c) Write the `addTour` method.

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.

Class information for this question

```
public class Tour
private int actDate
private int startTime
private int endTime
private String activity

Tour(int actDate, int startTime, int endTime,
String activity)
public int getActDate()
```

```
    public int getStartTime()
    public int getEndTime()
    public String getActivity()

    public class TravelPlan
    private String destination;
    private ArrayList <Tour> plans;

    public TravelPlan(String destination)
    public boolean addTour(Tour t)
    public boolean checkForConflicts(Tour t)
```

4. This question involves the implementation of a class seating chart. A `SeatingChart` object will represent a two-dimensional string array. The number of rows and columns for the array will be sent as parameters, as well as a one-dimensional array of type `Name`. You may assume there will be enough rows and columns to accommodate all the entries from the array.

   The declaration of the `Name` class is shown.

```
public class Name
{
     private String lastName;
     private String firstName;

     Name(String lName, String fName){<implementation not shown>}
     public String getLastName() {return lastName;}
     public String getFirstName() {return firstName;}
)
```

   A partial declaration of the `SeatingChart` class is shown below.

```
public class SeatingChart {
     private String [][] chart;
```

```
/** Constructs a SeatingChart having r rows and c columns. All elements contained in the
 *  names array should be placed randomly in the chart array using the format:
 *  lastName, firstName (e.g. Jolie, Angelina).
 *  Any locations not used in the chart should be
 *  initialized to the empty string.
*/
    SeatingChart(Name[] names, int rows, int cols){

        /* to be implemented in part (a) */

    }

/** Returns a string containing all elements of the chart array in row-major order.
 *  The method should return a string containing all the elements in the chart array.
 *  The method padWithSpaces should be called on each
 *  element of the chart before it is added to the string to ensure each name will be
 *  printed with the same length.
 *  Each row of the chart should be separated by a line break.
*/
    public String toString() {

            /* to be implemented in part (b) */

    }

/** Pads a string with spaces to ensure each string is exactly 35 characters long. */

    private String padWithSpaces(String s) {
        String str = s;
        for (int a = s.length(); a < 35; a++) {
            str += " ";
        }
```

```
            return str;
        }
    }
```

The following table contains sample code and the expected results.

Statements and Expressions	Value Returned / Comment
`SeatingChart msJones = new SeatingChart(theNames, 4, 3);`	(no value returned) A two-dimensional array is initialized with 4 rows and 3 columns. Every element in `theNames` is placed randomly in the chart in the following format: `lastname, firstname` (e.g., `Washington, George`). Empty string is placed in any unused locations.
`System.out.println(msJones.toString);`	Prints the names in the chart in row-major order. See example below:

```
Miller, Minnie      Fitzgerald, Fred      Dade, Ali
Indigo, Inde        Banner, Boris         Lane, Lois
Titon, Tim          Robilard, Robbie      Brne, Jane
Georgian, Greg
```

(a) Write the `SeatingChart` constructor.

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.

Class information for this question

```
public class Name
    private String lastName;
    private String firstName;

    Name(String lName, String fName)
    public String getLastName() {return
    lastName;}
    public String getFirstName() {return
    firstName;}

public class SeatingChart
    private String [][] chart;

    SeatingChart(Name[] names, int rows, int
    cols)
    public String toString()
    private String padWithSpaces(String s)
```

(**b**) Write the `SeatingChart toString()` method.

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.

Class information for this question

```
public class Name
    private String lastName;
    private String firstName;

    Name(String lName, String fName)
    public String getLastName() {return
    lastName;}
    public String getFirstName() {return
    firstName;)

public class SeatingChart
    private String [][] chart;

    SeatingChart(Name[] names, int rows, int
    cols)
    public String toString()
    private String padWithSpaces(String s)
```

STOP

END OF EXAM

Practice Test 1: Diagnostic Answer Key and Explanations

Click here to download a PDF of Diagnostic Answer Key Step 1.

PRACTICE TEST 1: DIAGNOSTIC ANSWER KEY

Let's take a look at how you did on Practice Test 1. Follow the three-step process in the diagnostic answer key below and read the explanations for any questions you got wrong, or you struggled with but got correct. Once you finish working through the answer key and the explanations, go to the next chapter to make your study plan.

Check your answers.

Section I: Multiple Choice			
Q #	**Ans.**	**✔**	**Chapter #, Title**
1	E		3, Objects & Primitive Data
2	D		3, Objects & Primitive Data 4, The Math Class
3	D		4, The String Class
4	A		4, The String Class 3, Objects & Primitive Data
5	C		4, The String Class
6	E		4, The String Class 3, Objects & Primitive Data
7	B		5, The If Statement
8	A		5, The If Statement 4, The String Class
9	E		5, The If Statement
10	C		5, The If Statement

11	C		6, The While Statement
12	D		6, The For Statement
13	A		6, The For Statement 5, The If Statement
14	C		3, Objects & Primitive Data 6, The For Statement
15	B		6, The For Statement
16	B		6, The While Statement 4, The String Class
17	D		6, The For Statement
18	C		7, Composition
19	B		7, Composition
20	D		7, Design & Structure
21	B		8, Primitives & Objects
22	E		8, Primitives & Objects
23	B		8, Primitives & Objects
24	D		8, Searches
25	A		8, Primitives & Objects
26	C		8, Primitives & Objects
27	E		9, Lists & ArrayLists
28	A		9, Lists & ArrayLists
29	B		6, Lists & ArrayLists
30	D		9, Lists & ArrayLists
31	D		10, 2D Arrays
32	B		10, 2D Arrays

33	B		11, Lists & ArrayLists
34	D		11, Lists & ArrayLists
35	C		11, Lists & ArrayLists
36	D		11, Lists & ArrayLists
37	E		11, Lists & ArrayLists
38	E		12, Recursion
39	D		12, Recursion
40	A		12, Recursively Traversing Arrays

Section II: Free-Response			
Q #	**Ans.**	**✔**	**Chapter #, Title**
1a	See Explanation		4, The Math Class
1b	See Explanation		6, The For Statement 5, The If Statement 4, The Math Class
2	See Explanation		7, Design & Structure 3, Objects & Primitive Data
3a	See Explanation		9, Lists & ArrayLists 7, Design & Structure
3b	See Explanation		9, Lists & ArrayLists 5, The If Statement 7, Methods
3c	See Explanation		9, Lists & ArrayLists 7, Methods
4a	See Explanation		10, 2D Arrays 8, Primitives & Objects 3, Objects & Primitive Data

4b	See Explanation		10, 2D Arrays

STEP 2»

Tally your correct answers from Step 1 by chapter. For each chapter, write the number of correct answers in the appropriate box. Then, divide your correct answers by the number of total questions (which we've provided) to get your percent correct.

CHAPTER 3 TEST SELF-EVALUATION

CORRECT ANSWERS ☐ / 7 # TOTAL QUESTIONS = ☐ % PERCENT CORRECT

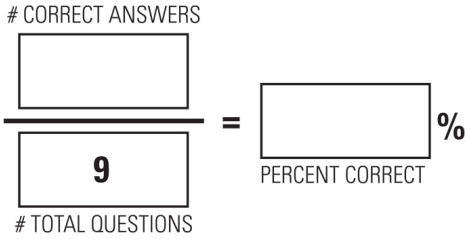
CHAPTER 4 TEST SELF-EVALUATION

CORRECT ANSWERS ☐ / 9 # TOTAL QUESTIONS = ☐ % PERCENT CORRECT

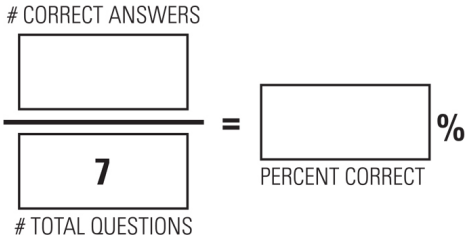
CHAPTER 5 TEST SELF-EVALUATION

CORRECT ANSWERS ☐ / 7 # TOTAL QUESTIONS = ☐ % PERCENT CORRECT

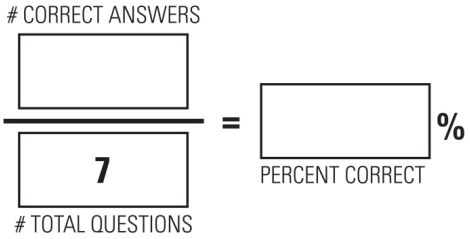
CHAPTER 6 TEST SELF-EVALUATION



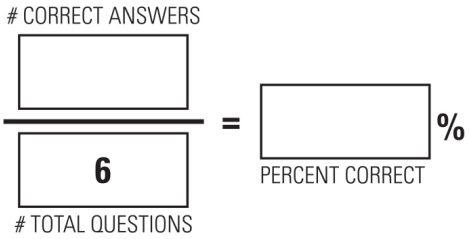
CHAPTER 7 TEST SELF-EVALUATION



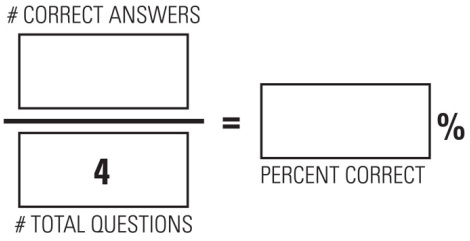
CHAPTER 8 TEST SELF-EVALUATION



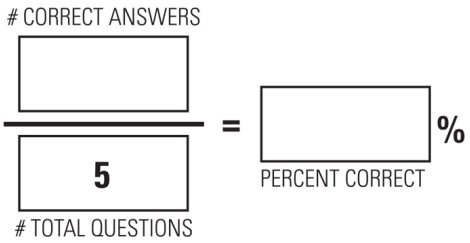
CHAPTER 9 TEST SELF-EVALUATION



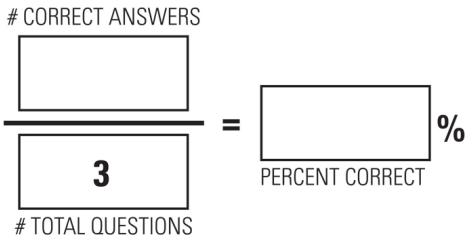
CHAPTER 10 TEST SELF-EVALUATION



CHAPTER 11 TEST SELF-EVALUATION



CHAPTER 12 TEST SELF-EVALUATION



Use the results above to customize your study plan. You may want to start with, or give more attention to, the chapters with the lowest percents correct.

PRACTICE TEST 1 EXPLANATIONS

Section I: Multiple-Choice Questions

1. **E**

   Modulus division and division have the same order of precedence. Going from left to right, modulus `(%)` is first: `6 % 12` is `6`. Division (/) is next and will be handled as integer division, since both terms of the operation are integers: `6 / 4` is `1`. Finally, do the addition: `4 + 1 = 5`. The correct answer is (E).

2. **D**

   Anytime a `double` data type is used in an operation, the result will yield a `double`. In (A) `(1.0 * 2)`, (B) `(2.0)`, and (C) `((double) 2)`, the numerators are all `2.0`. Choice (E) also yields `2.0`, since the `Math.sqrt` method returns a `double`. Choice (D) attempts to cast to `double` too late. The expression inside the parentheses `(2 / 10)` yields `0` before it can be cast to a `double`.

3. **D**

   First off, every string literal will be enclosed in quotation marks `("")`. Next, to print a character that serves as a control character with specific meanings in Java, characters like `\, “,` or `n` to indicate a new line, each character will have to be preceded by its own `\`. Thus, to print “`Friends`”, each “ that's printed will require its own `\`. Choices (A), (B), and (C) are missing the backslashes. Choice (E) has too many backslashes and will give a compiler error. Choice (D) is the correct answer because a backslash is used to indicate each control break.

4. **A**

The `String` class is immutable. Without additional assignment statements to change the values of `animal1` and `animal2`, they will retain the values assigned in the first two lines.

5. **C**

Choices (B), (D), and (E) all pass `String, String, int, int` as arguments to the second `Constellation` constructor. Choice (A) passes two strings to the first constructor. Choice (C) is the correct answer, as a double cannot be passed to a parameter of type `int` because there may be a loss of precision.

6. **E**

Segments I and II will use an `int` parameter to update the instance field(s) of type `int`. Segment III will cast the `double` to `int` before updating the instance field of type `int`. There may be a loss of precision, but it would be a logic error, not a compiler error. The correct answer is (E), as all options will compile correctly.

7. **B**

Trace the code:

```
int x = 10;
int y = 5;

if (x == 10)            //x is 10 so follow this branch
{
     if (y <= 5)     //y is 5 so follow this branch, add 1 to
     y, it is now 6
        y++;
     else if (y < 4)
             x = 3;
          else
```

```
            y += 6;
}                       //the first if statement is complete
if (y > 5)              //y is 6, so follow this branch
{
     if (x != 10)    //x is 10, so skip to the else
     {
          x = 0;
          y = 0;
     }
     else               //follow this branch, assign –5 to x
          x = –5;
}
                  //Thus, x = –5 and y = 6
```

The correct answer is (B).

8. **A**

The rules of `compareTo` are as follows: if `string1.compareTo(string2) < 0`, then the strings are in lexicographical order, whereas if `string1.compareTo(string2) > 0`, then the strings are in reversed order.

```
1 String word1 = “frog”;
2 String word2 = “dog”;
3 String word3 = “cat”;
4
5 if (word1.compareTo(word2) < 0)    //frog does not come before
  dog, skip to the else
6   if (word2.compareTo(word3) < 0)
7     System.out.println(word1 + “ ” + word2 + “ ” + word3);
8   else
9     System.out.println(word1 + “ ” + word3 + “ ” + word2);
10 else                                  //skip to here
11   if (word1.compareTo(word2) > 0) //frog comes after dog, so
  follow this branch
```

```
12    if (word2.compareTo(word3) < 0) //dog does not precede
  cat, skip to the else
13      System.out.println(word1 + " " + word2 + " " + word3);
14    else
15      System.out.println(word1 + " " + word3 + " " + word2);
  //frog cat dog
16  else
17    if (word2.equals(word3))
18      System.out.println("all the words are the same");
19    else
20      System.out.println("word1 and word2 are duplicates");
```

The correct answer is (A).

9. **E**

The following is given: `temp = 90` and `cloudy = false`. Segment I is evaluated as `true: temp >= 90 (true) && !cloudy(true)`. Both sides of the `&&` are true, so the entire condition is true. Option II is evaluated as `true`: De Morgan's Law can be used to simplify the `!()`. The simplified version is `temp <= 90 && !cloudy` —which are both true, so the entire condition is true. Segment III is also evaluated as `true`. Again, De Morgan's Law can be used to simplify the `!()`. The simplified version is `temp <= 90 || cloudy`. Since the `temp` is 90, the first condition is true. By short-circuit, the entire condition is true. The correct answer is (E).

10. **C**

Line 3 assigns `dog2`'s object reference to `dog1`. These two object variables are now pointing at the same object, the contents of which is "`Beagle`". Thus, the result of `if (dog1 == dog2)` on line 6 is `true`. Line 4 creates another object whose contents are "`Beagle`". Thus, the result of `if (dog1 == dog3)` on line 11 is `false`. The `==` is comparing whether the variables refer to the same object, not whether the content of the objects is the same.

The result of `if (dog1.equals(dog3))` on line 16 is `true`. The method `.equals` compares the contents of the two objects: they both contain "`Beagle`". The correct answer is (C).

11. **C**

Choice (A) starts at `0` and will decrement to a negative index, causing an out of bounds exception. Choices (B) and (D) start the index at `str1.length`, which is out of bounds. The last character in a string should be referenced by `length - 1`. Choice (E) correctly starts at `length - 1`; however, the loop only continues while the index is greater than 0, missing the first character of `str1`. The correct answer is (C).

12. **D**

Analytical problems of this type are more easily solved by selecting a value and testing the results. In this case, substitute a small number such as `3` for `n`, and then trace the code. The outer loop executes from `1` to `2`, which is 2 times. The inner loop will execute from `1` to `3`, which is 3 times. The code inside the loops is simply counting by 1. The inner loop will be executed (2 times 3) 6 times, thereby adding `6` to `count`.

Now, substitute `3` for `n` in all the possible answers.

	Expression	**Result**
(A)	3^3	27
(B)	$3^2 - 1$	8
(C)	$(3 - 1)^2$	4
(D)	$3(3 - 1)$	6
(E)	3^2	9

Thus, the answer to this problem is (D), `n(n - 1)`. Analytically, you could have looked at the first loop processing from `1` to `n - 1` and the second loop processing from `1` to `n`, and made the same assessment.

13. **A**

Choice (E) is eliminated with short-circuit. Line 4 looks to determine whether `x != 0`, but it IS `0`, so logic immediately branches to the `else` statement on line 7. Variable `x` is initialized to `0`, and `j` is initialized to `1`, so line 7 multiplies `j (1)` times `x (0) = 0` and prints the result. This eliminates (C) and (D). Both (A) and (B) are all zeroes, so the question becomes, how many `0`s will be printed? Line 2 specifies `j` will start at `1` and end at `3`, thus printing three `0`s. The correct answer is (A).

14. **C**

The loop located at lines 4–7 prints `symbol` (*) 5 times.

Line 8 is a control break to the next line.

The loop located at lines 9–16 is executed 5 times. The loop within at lines 11–14 prints `5 – j` spaces, so the first time through it will print 4 spaces, next time 3 spaces, and so on. (Note: this eliminates all answers except for (C).) After the spaces are printed on each line, a single `symbol` (*) is printed with `println` (which will then move to the next line).

The loop at 17–20 is the same as the first loop, printing `symbol` (*) 5 times. The correct answer is (C).

15. **B**

`i` will have only the value `1`, `j` will range from `1` to `3`, and `k` will range from `1` to `3`. The three variables will be multiplied by each

other and then added to the sum. The results will look like this:

```
i * j * k
1 * 1 * 1 = 1
1 * 1 * 2 = 2
1 * 1 * 3 = 3
1 * 2 * 1 = 2
1 * 2 * 2 = 4
1 * 2 * 3 = 6
1 * 3 * 1 = 3
1 * 3 * 2 = 6
1 * 3 * 3 = 9
```

The sum of which is `36`.

The correct answer is (B).

16. **B**

The `substring()` method has two parameters. The first specifies where to start, the second how far to go (up to but NOT including).

The outer loop at lines 3–12 is controlled by `j`. `j` starts off at `0`, eventually ending at `2`.

The inner loop at lines 5–10 is controlled by `k`. `k` starts off at `3` and will execute as long as it is greater than `j`.

The first time through the outer loop the following will be printed:

`s.substring(0, 3)` prints `map`
`s.substring(0, 2)` prints `ma`
`s.substring(0, 1)` prints `m`

The second time through the outer loop the following will be printed:

`s.substring(1, 3)` prints `ap`
`s.substring(1, 2)` prints `a`

The final time through the outer loop the following will be printed:

`s.substring(2, 3)` prints `p`

The correct answer is (B).

17. **D**

Once again, it is helpful to choose a value for `n` to analyze the code. Choosing `3` for `n`, analyze the code.

I—Each time through the loop, factorial will be multiplied by `3`. This does not follow the definition of a factorial. Eliminate (A) and (E), which include I.

II—The loop is controlled by `j`, which will range from `1` to `n`, in this case `3`. Each time through the loop, factorial is multiplied by `j`, thereby producing a result of `1 × 2 × 3`, which is correct. Eliminate (C).

III—A recursive solution that sends `n (3)` to the function
First pass is `f(3) -> 3 * f(2)`
Second pass is `f(2) -> 2 * f(1)`
Final pass is `f(1) -> 1 3 × 2 × 1` will yield `6` as expected. Eliminate (B).

The correct answer is (D), as only II and III will work.

18. **C**

When a local variable is created, it is used instead of the instance variable. When the constructor is invoked, line 28 does not update the instance variable price. Without specifying `this.price = price`, the local parameter is assigned the same value it already holds. Thus, (D) and (E) are eliminated. Choice (A) is eliminated because the `toString` method has been defined in the `Tile` class to print the instance variables (not the object reference). The `chgMaterial(mat)` method at line 30 also

updates a local variable rather than the instance variable, eliminating (B). The correct answer is (C).

19. **B**

If a print statement is passed an object, its `toString()` method will be invoked. This eliminates all answers except (B), which is the correct answer.

20. **D**

A static variable would be used for something that would belong to the entire class. Since inventory needs to exist for each style, it cannot be static, but it must be an instance of the class, eliminating (A) and (B). Choice (C) is eliminated because the keyword `final` is used only for constants that do not change value, but the owner has also asked for a method to change the value. Since `styleNumber` is an instance field, it implies that a separate instance is created for each style. Thus an array is not needed, eliminating (E). The correct answer is (D).

21. **B**

The array is initialized as `{11, 22, 33, 44, 55, 66};`

First pass:
```
nums[nums[0] / 11] = nums[0];
nums[11 / 11] = nums[0];
nums[1] = nums[0];
```
The array is now: `{11, 11, 33, 44, 55, 66};`

Second pass:
```
nums[nums[1] / 11] = nums[1];
nums[11 / 11] = nums[1];
nums[1] = nums[1];
```
The array is unchanged: `{11, 11, 33, 44, 55, 66};`

Third pass: `nums[nums[2] / 11] = nums[2];`

```
nums[33 / 11] = nums[2];
nums[3] = nums[2];
```
The array is now: `{11, 11, 33, 33, 55, 66};`

Fourth pass: `nums[nums[3] / 11] = nums[3];`

```
nums[33 / 11] = nums[3];
nums[3] = nums[3];
```
The array is unchanged: `{11, 11, 33, 33, 55, 66};`

Fifth pass: `nums[nums[4] / 11] = nums[4];`

```
nums[55 / 11] = nums[4];
nums[5] = nums[4];
```
The array is now: `{11, 11, 33, 33, 55, 55};`

Sixth pass: `nums[nums[5] / 11] = nums[5];`

```
nums[55 / 11] = nums[5];
nums[5] = nums[5];
```
The array is unchanged: `{11, 11, 33, 33, 55, 55};`

The correct answer is (B).

22. **E**

Line 14 assigns the `arr1` object reference to `arr2` object reference. Thus, both variables are now pointing to the exact same array in memory.

The loop at lines 17–18 is the only code that modifies the array.

both `arr1` and `arr2: {1, 2, 3, 4, 5, 6}; last = 5`

```
arr2[i] = arr1[last - i];
```

first pass: `arr2[0] = arr1[5 - 0]; {6, 2, 3, 4, 5, 6}`
second pass: `arr2[1] = arr1[5 - 1]; {6, 5, 3, 4, 5, 6}`
third pass: `arr2[2] = arr1[5 - 2]; {6, 5, 4, 4, 5, 6}`
fourth pass: `arr2[3] = arr1[5 - 3]; {6, 5, 4, 4, 5, 6}`
fifth pass: `arr2[4] = arr1[5 - 4]; {6, 5, 4, 4, 5, 6}`
last pass: `arr2[5] = arr1[5 - 5]; {6, 5, 4, 4, 5, 6}`

The correct answer is (E).

23. **B**

The `for` loop on line 29 creates a local variable named `element` which will hold each value of `arr3` without having to use an index. Modifying this local variable does not modify the individual contents within the array. The loop multiplies each element by 2, printing it as it does so.

```
2, 4, 6, 8, 10, 12
```

The loop at line 36 prints the contents of the array, which remain unchanged:

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

The correct answer is (B).

24. **D**

Since index `i` is assigned to variables `a` and `b`, it is locations that are being printed. This eliminates (A). Scan the remaining answers and make a chart to help you understand the possibilities.

	Location of:	**Location of:**
(B)	Smallest integer	Largest integer
(C)	Smallest non-negative integer	Largest non-negative integer
(D)	Smallest integer	Largest non-negative integer
(E)	Smallest non-negative	Largest integer

	integer	

The variable `holdSmallest` is initialized with `Integer.MAX_ VALUE`, which is the largest integer an `int` field may hold. Thus, the code will work to find the smallest number in the array even if it is a negative number. This eliminates (C) and (E). The variable `holdLargest` is initialized to `0`, so when looking for a larger integer, it will only be replaced if it is larger than `0`, or in other words, a non-negative integer. This eliminates (B). The correct answer is (D).

25. **A**

Choice (B) is eliminated because there is no increment to variable `i`. Choice (C) is eliminated because without an index, it implies the entire array (not each element) is being added to `sum` over and over. Choice (D) cannot use `element`, because it will contain the contents of a location within the array, rather than a location. Choice (E) uses the variable name of the array as the index. Choice (A) is correct because it uses the temporary variable `element`, which will actually hold the contents of each location within the array.

26. **C**

Examining the code of `swap1`, you can see it will work only if the arrays are the same length. There is no accommodation for one array being longer than the other. In fact, if `a1` is longer, there will be an out of bounds error on the second array. This eliminates (A), (D), and (E). The code of `swap2` does not work. Array variables hold a reference to the array, not to the actual elements. This eliminates (B). The correct answer is (C).

27. **E**

Segment I declares an `ArrayList` of type `String` and then adds "`4.5`", which is a `String`. It is correct. Eliminate (D), which does not include I.

Segment II declares an `ArrayList` of type `Integer` and then casts `4.5` to an `int` before adding it to the `ArrayList`, which is acceptable. It is correct. Eliminate (A) and (C).

Segment III declares an `ArrayList` variable and then completes the declaration of the `ArrayList` as type `double` on the next line. It then adds a double to the `ArrayList`, which is correct. Eliminate (B).

The correct answer is (E).

28. **A**

The first loop loads the contents of the array into the `ArrayList`. The next loop begins to remove elements if those elements are even. The loop will continue to run until it reaches the size of the `ArrayList`. As elements of the `ArrayList` are removed, the size will decrease, so there is no risk of going out of bounds. Eliminate (E). However, the index `i` will occasionally skip elements because of the renumbering that takes place.

```
[2, 4, 6, 7, 8, 10, 11];
```

`i = 0` The `2` is even, so it is removed; the array is now

```
[4, 6, 7, 8, 10, 11];
```

`i = 1` Notice the `4` will now be skipped. The `6` is even, so it is removed; the array is now

```
[4, 7, 8, 10, 11];
```

`i = 2` The `8` is even, so it is removed; the array is now

```
[4, 7, 10, 11];
```

`i = 3` The `10` has been skipped. The `11` is odd, so the array stays the same:

```
[4, 7, 10, 11];
```

The correct answer is (A).

29. **B**

The size of the array is `5`, so `size - 1` is `4`. The outer loop executes 4 times `(0–3)`.

`size - 2` is `3`. The inner loop executes 3 times `(0–2)`.

Since line 12 is executed every time the inner loop is executed, it will be executed (4)(3) = 12 times. The correct answer is (B).

30. **D**

The inner loop does not go far enough to process the entire array. `size` is `5`, and `size - 1` is `4`, so the index can only be less than `4`, stopping at index `3`. The last entry in the `ArrayList` will never be sorted. The sort makes 4 passes through the `ArrayList`. The passes will look as follows:

```
0    [6, 21, 2, 8, 1]

1    [6, 2, 21, 8, 1]

2    [6, 2, 8, 21, 1]

3    [2, 6, 8, 21, 1]
```

The correct answer is (D).

31. **D**

The array is printed in column-major order. The outer loop runs from 0 to row length – 1 (the number of columns). The inner loop runs from 0 to the length of the array (which means the number of rows).

The original array is

```
1  2  3  4

5  6  7  8
```

The outer loop starts with column 0, prints `[0] [0]: 1  [1] [0]: 5`

The outer loop increments to column 1 `[0] [1]: 2  [1] [1]: 6`

The outer loop increments to column 2 `[0] [2]: 3  [1] [2]: 7`

The outer loop increments to column 3 `[0] [3]: 4  [1] [3]: 8`

The correct answer is (D).

32. **B**

Segment III will go out of bounds. The `r` (rows) will iterate as many times as there are columns. If there are fewer rows than columns, the index will go out of bounds. The correct answer is (B).

33. **B**

Since `name` and `weight` are instance variables in the `Percussion class`, values for those variables should be passed while calling `super`. The call to `super` must be the first line in a method. Thus, (A), (D), and (E) are eliminated. The assignment statement of `numberOfKeys` is reversed in (C). The local variable is being initialized by the instance field. The correct answer is (B).

34. **D**

The variable `numberOfKeys` is not visible outside the `Xylophone` class. Choices (A) and (B) are simply creating arrays of `Xylophone` objects. Choice (C) creates a `xylophone` object and then uses the proper accessor method to print the number of keys. Choice (E) declares a variable for an array of type `Drums`. Choice (D) attempts to print a private instance variable without using an accessor method. It will not comply, so the correct answer is (D).

35. **C**

The accessor method `getWeight()` will return the weight of each instance so that they can be compared. Choice (A) is incorrect because the `weight` field is not visible. Choice (B) is not correct because `weight()` is not a defined method. Choice (D) is not correct because not only is `weight` not visible, but `.equals` is not used to compare primitive types. Choice (E) is incorrect because (C) compares the fields correctly. The correct answer is (C).

36. **D**

Use the IS-A relationship to check the solutions:

(A)—`SportingDog` is a `Dog` (yes)

(B)—`Retriever` is a `Dog` (yes)

(C)—`Retriever` is a `Sporting Dog` (yes)

(D)—`Dog` is a `Sporting Dog` (no, the relationship is the opposite: not all dogs are sporting dogs)

(E)—`Retriever` is a `Retriever` (yes)

The correct answer is (D).

37. **E**

The `Retriever toString()` method is invoked first, returning `type: Labrador + super.toString()`.

No `toString()` method is found in `SportingDog`, but a `toString()` method is found in `Dog`, adding `color is: chocolate` to the print line.

The correct answer is (E).

38. **E**

Try substituting numbers for the variables. Try finding 3^2 by making `b = 3, x = 2`. The solution is found by multiplying 3×3.

The base case will be 3 (when the exponent is 1). This should imply that the `if` statement at line 3 should be

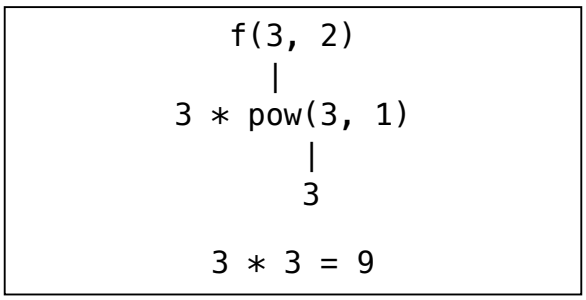
```
if (x == 1)   return b;
```

There is another error on line 6. Line 6 is using addition, when raising to a power is multiplying the base `x` times. Thus, the `+` sign should be changed to multiplication.

After making the changes in the code, it is advisable to test it to ensure it works:

```
          b = 3, x = 2

1 public double pow(double b, int x)
2 {
3 if (x == 1)
4   return b;
5 else
6   return b * pow(b, x - 1);
7 }
```

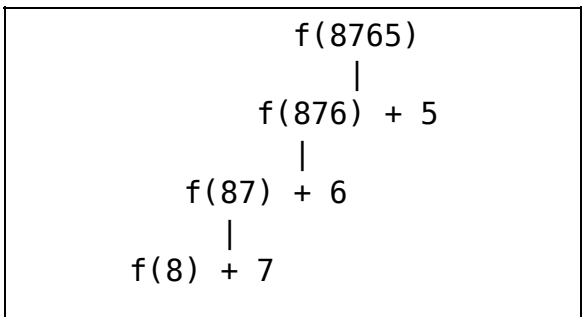


The correct answer is (E).

39. **D**

It is best to walk the code.

```
System.out.println(f(8765));
public static int f(int n)
{
     if (n == 0)
          return 0;
     else
          return f(n / 10) + n % 10;
}
```



```
        |
0 + 8

8 + 7 + 6 + 5 = 26
```

The correct answer is (D).

40. **A**

If `sought` is less than the element at index `mid`, the beginning of the array should be searched. The location of the middle of the array, `mid – 1`, should be assigned to `last`. If `sought` is greater than the element at index `mid, mid + 1` should be assigned to `first` so that the latter half of the array can be searched. This process should be repeated until `sought` is found. The correct answer is (A).

Section II: Free-Response Questions

1. `DiceSimulation`—Canonical Solution

(a)
```
public int roll() {

     return (int)(Math.random() * numFaces + 1);
}
```

(b)
```
public int runSimulation()
{
int die1 = 0;
int die2 = 0;
int countDouble = 0;
for (int i = 0; i < numSampleSize; i++) {
     die1 = roll();
     die2 = roll();
     if (die1 == die2) {
          countDouble++;
```

```
        }
    }
    return (int)((1.0 * countDouble / numSampleSize) * 100);
    }
```

`DiceSimulation` Rubric

Part (**a**)

+3	`roll` method	
	+1	`Math.random()` or the `Random` class is used
	+1	multiplied by `numFaces + 1`
	+1	result of computation is cast to `int` appropriately and returned

Part (**b**)

+6	`runSimulation` method	
	+1	local variables are declared and initialized for the two dice
	+1	`roll` is used to give the dice values
	+1	a loop is used to execute sample size times (no more, no less)
	+1	the values of `die1` and `die2` are compared with ==, doubles are counted appropriately
	+1	the percentage of doubles is calculated (avoiding integer division), multiplied by 100
	+1	percentage is returned as an `int`

2. `CalorieCount`—Canonical Solution

```
public class CalorieCount {
    private int numCaloriesLimit;
    private int numCaloriesIntake;
```

```
    private int gramsProtein;
    private int gramsCarbohydrate;
    private int gramsFat;

    public CalorieCount (int numCal) {
        numCaloriesLimit = numCal;
        numCaloriesIntake = 0;
        gramsProtein = 0;
        gramsCarbohydrate = 0;
        gramsFat = 0;
    }
    public void addMeal(int calories, int protein, int
    carbs, int fat) {
        numCaloriesIntake += calories;
        gramsProtein += protein;
        gramsCarbohydrate += carbs;
        gramsFat += fat;
    }
    public double getProteinPercentage() {
        return 4.0 * gramsProtein / numCaloriesIntake;
    }
    public boolean onTrack() {
        return numCaloriesIntake <= numCaloriesLimit;
    }
}
```

`CalorieCount` Rubric

+1 Declares all appropriate `private` instance variables

+2 Constructor

- +1 declares header: `public CalorieCount (int calorieLimit)`
- +1 uses parameters and appropriate values to initialize instance variables

+2 `addMeal` method

- +1 declares header: `public void addMeal(int calories, int protein, int carbs, int fat)`

+1 updates instance variables appropriately

+2 `getProteinPercentage` method

+1 declares header: `public double getProteinPercentage()`

+1 calculation and return: `return 4.0 * gramsProtein / numCaloriesIntake;`

+2 `onTrack` method

+1 declares header: `public boolean onTrack()`

+1 correctly returns `true` or `false`

e.g., `return numCaloriesIntake <= numCaloriesLimit;`

3.

(a)
```
TravelPlan(String destination){
        this.destination = destination;
        plans = new ArrayList<Tour>();
    }
```

(b)
```
public boolean checkForConflicts(Tour t) {
        for (int i = 0; i < plans.size(); i++)
        {
             if (t.getActDate() == plans.get(i).getActDate())
             {
                  int plannedStart =
                  plans.get(i).getStartTime();
                  int plannedEnd = plans.get(i).getEndTime();
                  if ((t.getStartTime() >= plannedStart &&
                  t.getStartTime() < plannedEnd) ||
                  (t.getEndTime() > plannedStart &&
                  t.getEndTime() < plannedEnd))
                       return true;
                  if (t.getStartTime() <= plannedStart &&
                  t.getEndTime() >= plannedEnd)
                       return true;
```

```
            }
        }
        return false;
    }
```

(c)
```
public boolean String addTour(Tour t) {
        if (checkForConflicts(t)) {
            return false;
        }
        plans.add(t);
        return true;
    }
```

`TravelPlan` Rubric

Part **(a)**

+3	Constructor	
	+1	constructor uses class name `TravelPlan`
	+1	updates `destination` instance field appropriately (uses `this`.)
	+1	creates `ArrayList` appropriately

Part **(b)**

+4	`checkForConflicts` method	
	+1	uses a loop to traverse every item in the `ArrayList` (no bounds errors)
	+1	uses `.get(index)` to access the object in the `ArrayList`
	+1	uses `getStartTime ()` and `getEndTime()` to access the private fields in the `Tour` object
	+1	uses appropriate logic to determine whether there is a time conflict on the same day; returns `true` if there is a conflict, `false` otherwise

Part (**c**)

+2 `addTour` method

- +1 calls `checkForConflict` method to determine whether there is a conflict (loses this point if it instead writes the logic to determine whether there is a conflict in this method), adds tour if there is no conflict
- +1 returns `true` if tour is added, or `false` if tour is not added

4. `SeatingChart` — Canonical Solution

(a)

```
SeatingChart(Name[] names, int r, int c)
{
     chart = new String[r][c];
     for (int i = 0; i < chart.length; i++)
     {
          for (int j = 0; j < chart[0].length; j++)
          {
               chart[i][j] = “”;
          }
     }
     int count = 0;
     int i = (int) (Math.random() * names.length);
     int row = i / c;
     int col = i % c;
     while (count < names.length) {
          while (!chart[row][col].equals(“”))
          {
               i = (int) (Math.random() * names.length);
               row = i / c;
               col = i % c;
          }
          chart[row][col]= names[count].getLastName() + “,
          ” +
names[count].getFirstName();
```

```
            count ++;
            }
    }
```

(b)

```
public String toString()
    {
         String str = “”;
         for (int a = 0; a < chart.length; a++) {
              for (int b = 0; b < chart[a].length; b++) {
                   str += padWithSpaces(chart[a][b]);
              }
              str += “\n”;
         }
         return str;
    }
```

`SeatingChart` Rubric

Part **(a)**

+6	Constructor	
	+1	chart is initialized using rows and columns passed in parameters
	+1	random numbers are generated in the correct range
	+1	a unique random number is used each time to place the name in the 2D array (duplicate names are avoided) and all names are placed (none are skipped)
	+1	row and column in the seating chart are derived properly from the random number
	+1	the name is stored in `chart` as a string (last name, (comma), first name), e.g., `Washington, George`

	+1	any unused spaces left in the array should be initialized to the empty string (not `null`)

Part (**b**)

+3	`toString` method	
	+1	builds a single string with all names from the 2D array, calling `padWithSpaces` to make all names an equal length
	+1	"`\n`" creates a line break after each row
	+1	returns a string

HOW TO SCORE PRACTICE TEST 1

Click here to download a PDF of How to Score Practice Test 1.

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))