U5-10_2

Name

1. Consider the following method, which implements a recursive binary search.

```
/** Returns an index in arr where the value x appears if x appears
*  in arr between arr[left] and arr[right], inclusive;
*  otherwise returns -1.
*  Precondition: arr is sorted in ascending order.
*                left >= 0, right < arr.length, arr.length > 0
*/
public static int bSearch(int[] arr, int left, int right, int x)
{
if (right >= left)
{
int mid = (left + right) / 2;
if (arr[mid] == x)
{
return mid;
}
else if (arr[mid] > x)
{
return bSearch(arr, left, mid - 1, x);
}
else
{
return bSearch(arr, mid + 1, right, x);
}
}
return -1;
}
```

The following code segment appears in a method in the same class as bSearch.

```
int[] nums = {0, 4, 4, 5, 6, 7};
int result = bSearch(nums, 0, nums.length - 1, 4);
```

What is the value of result after the code segment has been executed?

U5-10_2

Ⓐ 1

Ⓑ 2

Ⓒ 3

Ⓓ 4

Ⓔ 5

U5-10_2

2. Consider the following method, which implements a recursive binary search.

```
/** Returns an index in myList where target appears,
* if target appears in myList between the elements at indices
* low and high, inclusive; otherwise returns -1.
* Precondition: myList is sorted in ascending order.
* low >= 0, high < myList.size(), myList.size() > 0
*/
public static int binarySearch(ArrayList<Integer> myList,
int low, int high, int target)
{
int mid = (high + low) / 2;
if (target < myList.get(mid))
{
return binarySearch(myList, low, mid - 1, target);
}
else if (target > myList.get(mid))
{
return binarySearch(myList, mid + 1, high, target);
}
else if (myList.get(mid).equals(target))
{
return mid;
}
return -1;
}
```

Assume that inputList is an ArrayList of Integer objects that contains the following values.

```
[0, 10, 30, 40, 50, 70, 70, 70, 70]
```

What value will be returned by the call binarySearch(inputList, 0, 8, 70) ?

Ⓐ -1

Ⓑ 5

Ⓒ 6

Ⓓ 7

Ⓔ 8

3. Consider the following class definition.

```
public class Document
{
private int pageCount;
private int chapterCount;
public Document(int p, int c)
{
pageCount = p;
chapterCount = c;
}
public String toString()
{
return pageCount + " " + chapterCount;
}
}
```

The following code segment, which is intended to print the page and chapter counts of a Document object, appears in a class other than Document.

```
Document d = new Document(245, 16);
System.out.println( /* missing code */ );
```

Which of the following can be used as a replacement for /* missing code */ so the code segment works as intended?

(A) `d.toString()`

(B) `toString(d)`

(C) `d.pageCount + " " + d.chapterCount`

(D) `d.getPageCount() + " " + d.getChapterCount()`

(E) `Document.pageCount + " " + Document.chapterCount`

4. A two-dimensional array arr is to be created with the following contents.

```
boolean[][] arr = {{false, true, false},
{false, false, true}};
```

Which of the following code segments can be used to correctly create and initialize arr ?

(A)
```
boolean arr[][] = new boolean[2][3];
arr[0][1] = true;
arr[1][2] = true;
```

(B)
```
boolean arr[][] = new boolean[2][3];
arr[1][2] = true;
arr[2][3] = true;
```

(C)
```
boolean arr[][] = new boolean[3][2];
arr[0][1] = true;
arr[1][2] = true;
```

(D)
```
boolean arr[][] = new boolean[3][2];
arr[1][0] = true;
arr[2][1] = true;
```

(E)
```
boolean arr[][] = new boolean[3][2];
arr[2][1] = true;
arr[3][2] = true;
```

U5-10_2

5. Consider the following method countNegatives, which searches an ArrayList of Integer objects and returns the number of elements in the list that are less than 0.

```
public static int countNegatives(ArrayList<Integer> arr)
{
int count = 0;
for (int j = 0; j < arr.size(); j++)   // Line 4
{
if (arr.get(j) < 0)
{
count++;
}
}
return count;
}
```

Which of the following best explains the impact to the countNegatives method when, in line 4, j < arr.size() is replaced with j <= arr.size() - 1 ?

(A) It has no impact on the behavior of the method.

(B) It causes the method to ignore the last element in arr.

(C) It causes the method to throw an IndexOutOfBounds exception.

(D) It reduces the size of arr by 1 and the last element will be removed.

(E) It changes the number of times the loop executes, but all indexes in arr will still be accessed.

U5-10_2

6. Consider the following method findValue, which takes an ArrayList of String elements and a String value as parameters and returns true if the String value is found in the list and false otherwise.

```
public static boolean findValue(ArrayList<String> arr, String key)
{
for (int j = 0; j < arr.size(); j++)      // Line 3
{
if (arr.get(j).equals(key))
{
return true;
}
}
return false;
}
```

Which of the following best explains the impact to the findValue method when, in line 3, int j = 0 is replaced by int j = 1 ?

(A) It has no impact on the behavior of the method.

(B) It will cause the method to return a different result when the key value is not in the list.

(C) It will cause the method to return a different result when the key value is found only at the first index in the list.

(D) It will cause the method to return a different result when the key value is found only at the last index in the list.

(E) It will cause the method to throw an array index out of bounds exception.

U5-10_2

7. Consider the following method, inCommon, which takes two Integer ArrayList parameters. The method returns true if the same integer value appears in both lists at least one time, and false otherwise.

```
public static boolean inCommon(ArrayList<Integer> a, ArrayList<Integer>
b)
{
for (int i = 0; i < a.size(); i++)
{
for (int j = 0; j < b.size(); j++)     // Line 5
{
if (a.get(i).equals(b.get(j)))
{
return true;
}
}
}
return false;
}
```

Which of the following best explains the impact to the inCommon method when line 5 is replaced by for (int j = b.size() - 1; j > 0; j--) ?

(A) The change has no impact on the behavior of the method.

(B) After the change, the method will never check the first element in list b.

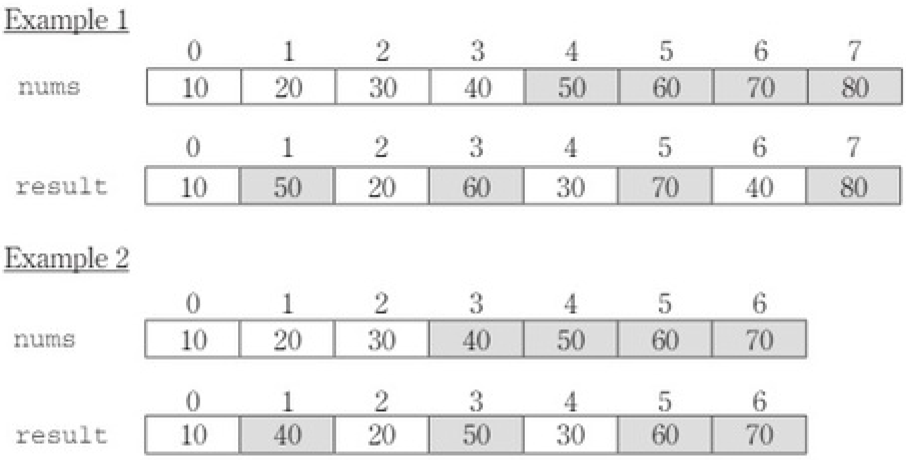
(C) After the change, the method will never check the last element in list b.

(D) After the change, the method will never check the first and the last elements in list b.

(E) The change will cause the method to throw an IndexOutOfBounds exception.

U5-10_2

8. Consider a shuffle method that is intended to return a new array that contains all the elements from nums, but in a different order. Let n be the number of elements in nums. The shuffle method should alternate the elements from nums [0] … nums[n / 2 – 1] with the elements from nums[n / 2] …nums[n – 1], as illustrated in the following examples.



Example 1

	0	1	2	3	4	5	6	7
nums	10	20	30	40	50	60	70	80

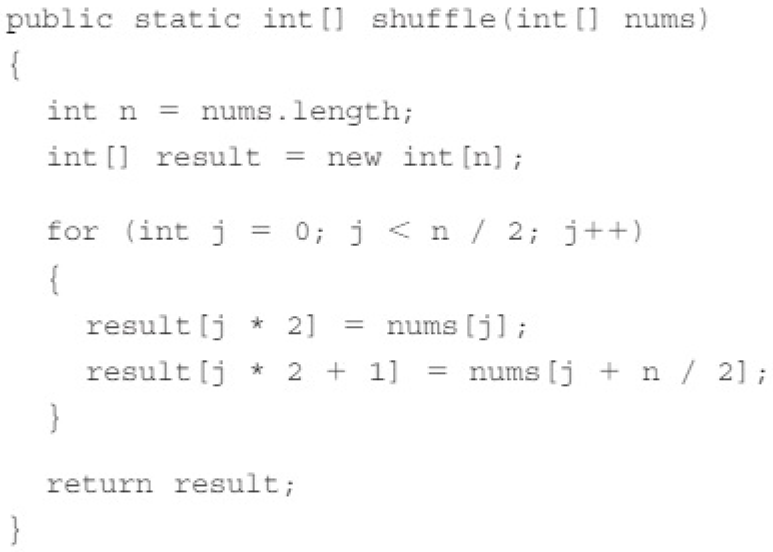
	0	1	2	3	4	5	6	7
result	10	50	20	60	30	70	40	80

Example 2

	0	1	2	3	4	5	6
nums	10	20	30	40	50	60	70

	0	1	2	3	4	5	6
result	10	40	20	50	30	60	70

The following implementation of the shuffle method does not work as intended.



```
public static int[] shuffle(int[] nums)
{
  int n = nums.length;
  int[] result = new int[n];

  for (int j = 0; j < n / 2; j++)
  {
    result[j * 2] = nums[j];
    result[j * 2 + 1] = nums[j + n / 2];
  }

  return result;
}
```

Which of the following best describes the problem with the given implementation of the shuffle method?

U5-10_2

Ⓐ Executing shuffle may cause an ArrayIndexOutOfBoundsException.

Ⓑ The first element of the returned array (result [0]) may not have the correct value.

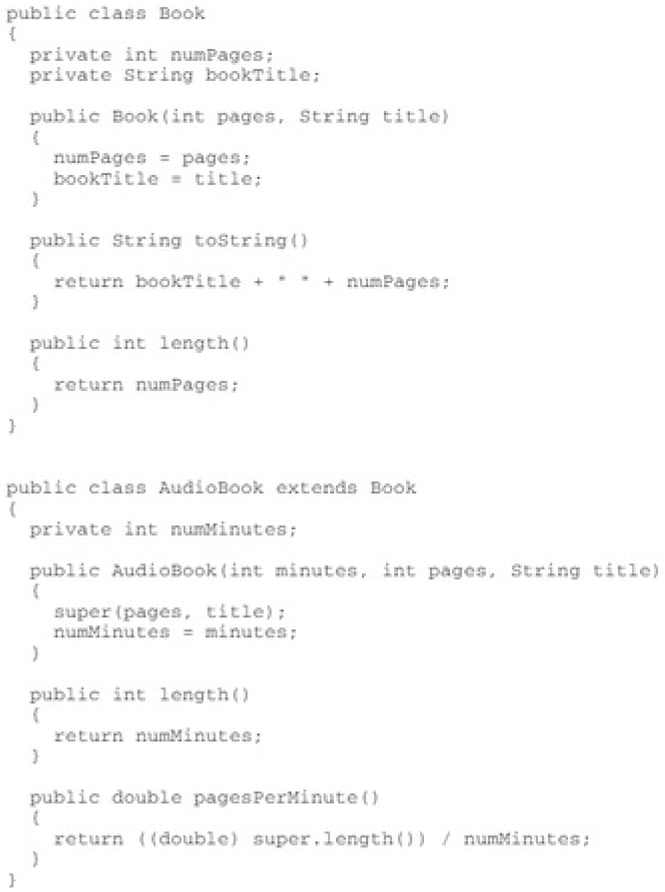
Ⓒ The last element of the returned array (result [result.length − 1]) may not have the correct value.

Ⓓ One or more of nums [0] … nums [nums.length / 2 − 1] may have been copied to the wrong position(s) in the returned array.

Ⓔ One or more of nums [nums.length / 2] … nums[nums.length − 1] may have been copied to the wrong position(s) in the returned array.

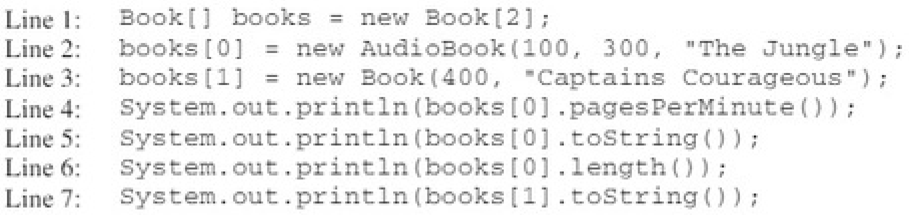
U5-10_2

9. Consider the following Book and AudioBook classes.



```
public class Book
{
  private int numPages;
  private String bookTitle;

  public Book(int pages, String title)
  {
    numPages = pages;
    bookTitle = title;
  }

  public String toString()
  {
    return bookTitle + " " + numPages;
  }

  public int length()
  {
    return numPages;
  }
}


public class AudioBook extends Book
{
  private int numMinutes;

  public AudioBook(int minutes, int pages, String title)
  {
    super(pages, title);
    numMinutes = minutes;
  }

  public int length()
  {
    return numMinutes;
  }

  public double pagesPerMinute()
  {
    return ((double) super.length()) / numMinutes;
  }
}
```

Consider the following code segment that appears in a class other than Book or AudioBook.



```
Line 1:  Book[] books = new Book[2];
Line 2:  books[0] = new AudioBook(100, 300, "The Jungle");
Line 3:  books[1] = new Book(400, "Captains Courageous");
Line 4:  System.out.println(books[0].pagesPerMinute());
Line 5:  System.out.println(books[0].toString());
Line 6:  System.out.println(books[0].length());
Line 7:  System.out.println(books[1].toString());
```

Which of the following best explains why the code segment will not compile?

U5-10_2

Ⓐ Line 2 will not compile because variables of type Book may not refer to variables of type AudioBook.

Ⓑ Line 4 will not compile because variables of type Book may only call methods in the Book class.

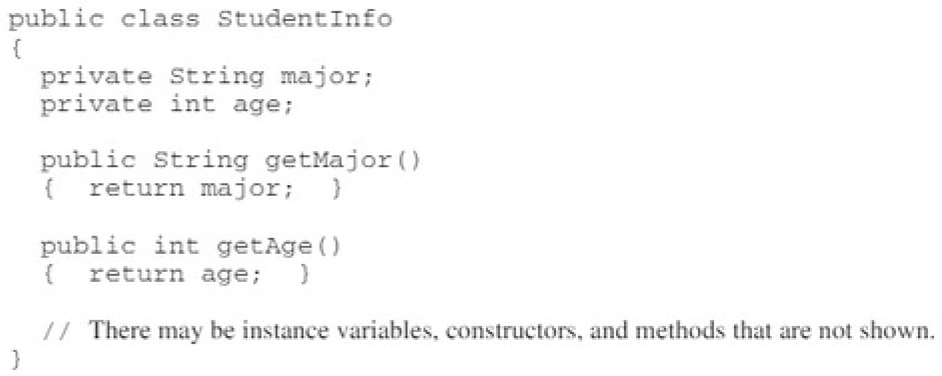
Ⓒ Line 5 will not compile because the AudioBook class does not have a method named toString declared or implemented.

Ⓓ Line 6 will not compile because the statement is ambiguous. The compiler cannot determine which length method should be called.

Ⓔ Line 7 will not compile because the element at index 1 in the array named books may not have been initialized.

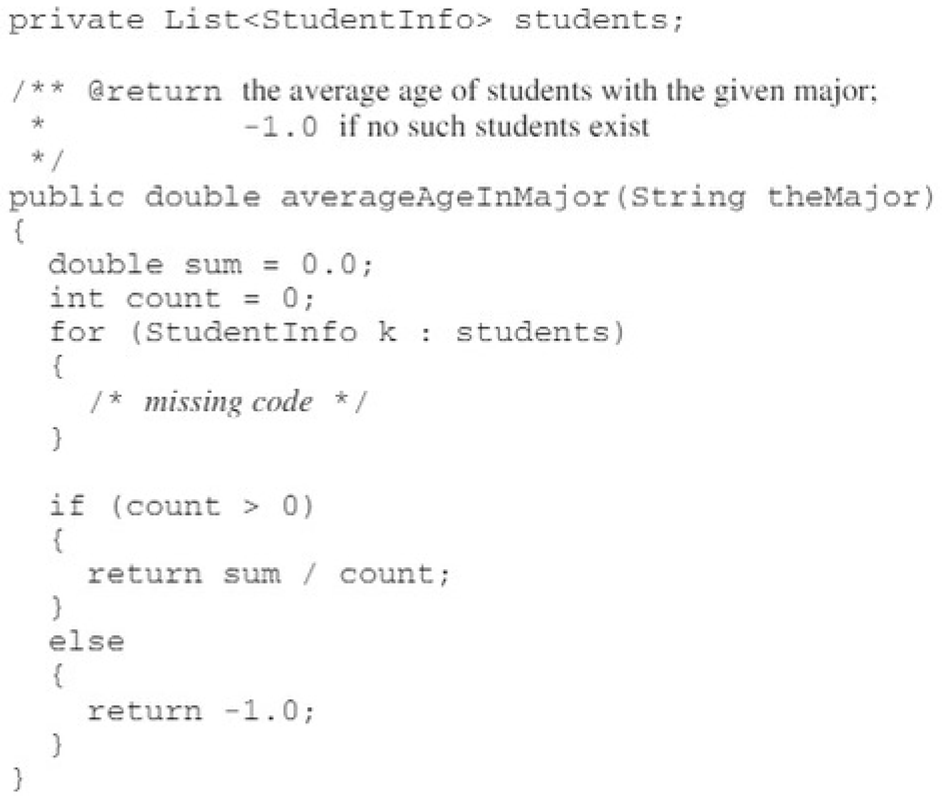
U5-10_2

10. Consider the following class declaration.



```
public class StudentInfo
{
  private String major;
  private int age;

  public String getMajor()
  {  return major;  }

  public int getAge()
  {  return age;  }

  // There may be instance variables, constructors, and methods that are not shown.
}
```

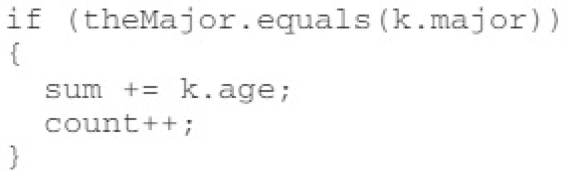
The following instance variable and method appear in another class.



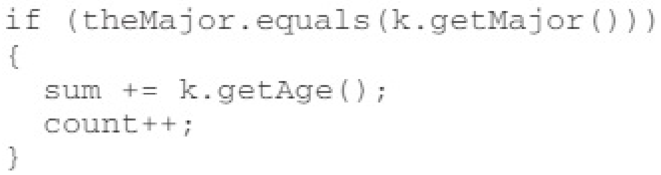
```
private List<StudentInfo> students;

/** @return  the average age of students with the given major;
 *           -1.0  if no such students exist
 */
public double averageAgeInMajor(String theMajor)
{
  double sum = 0.0;
  int count = 0;
  for (StudentInfo k : students)
  {
    /* missing code */
  }

  if (count > 0)
  {
    return sum / count;
  }
  else
  {
    return -1.0;
  }
}
```

Which of the following could be used to replace /* *missing code* */ so that averageAgeInMajor will compile without error?

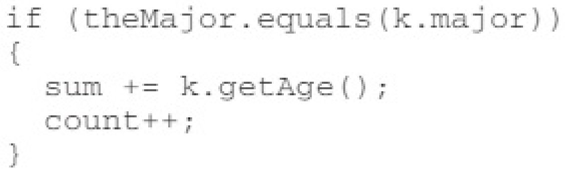
U5-10_2

Ⓐ 

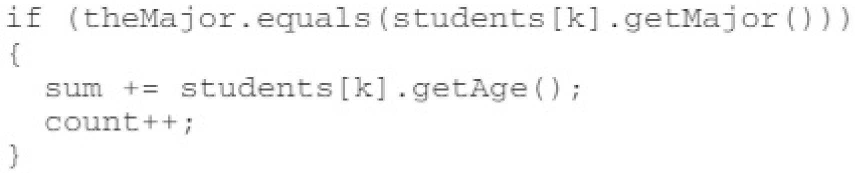
```
if (theMajor.equals(k.major))
{
  sum += k.age;
  count++;
}
```

Ⓑ 

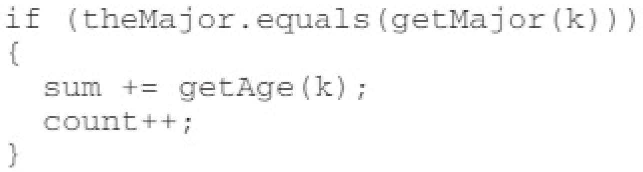
```
if (theMajor.equals(k.getMajor()))
{
  sum += k.getAge();
  count++;
}
```

Ⓒ 

```
if (theMajor.equals(k.major))
{
  sum += k.getAge();
  count++;
}
```

Ⓓ 

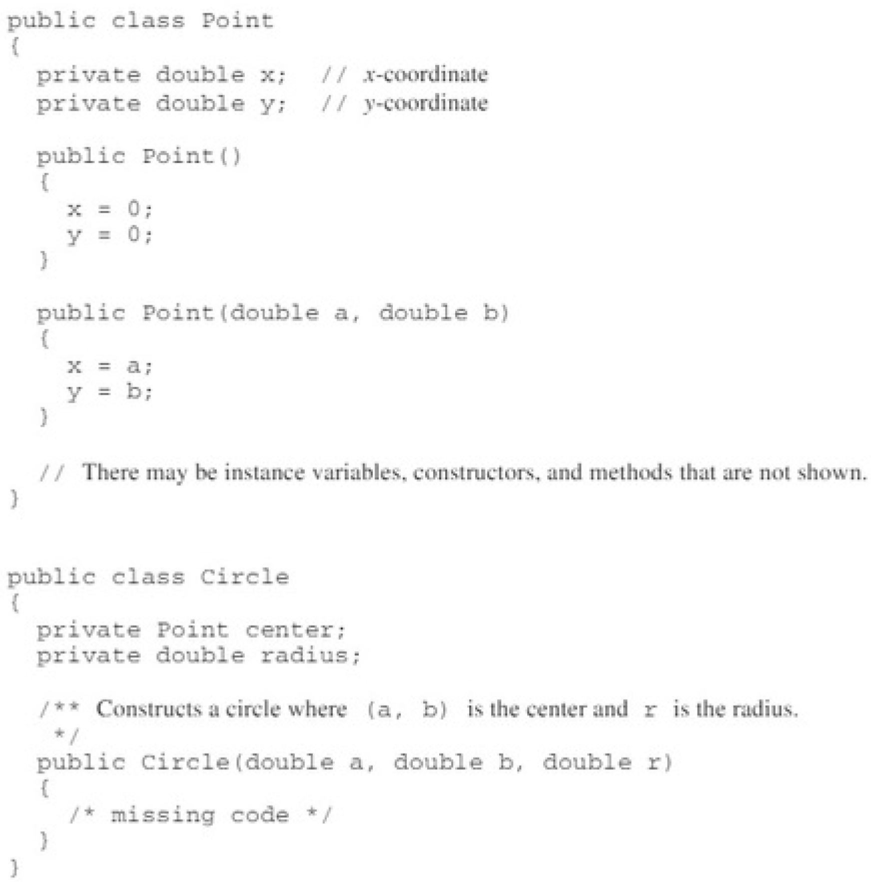
```
if (theMajor.equals(students[k].getMajor()))
{
  sum += students[k].getAge();
  count++;
}
```

Ⓔ 

```
if (theMajor.equals(getMajor(k)))
{
  sum += getAge(k);
  count++;
}
```

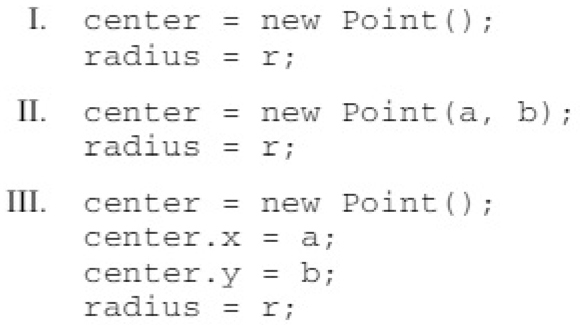
U5-10_2

11. Consider the following class declarations.



```
public class Point
{
  private double x;   // x-coordinate
  private double y;   // y-coordinate

  public Point()
  {
    x = 0;
    y = 0;
  }

  public Point(double a, double b)
  {
    x = a;
    y = b;
  }

  // There may be instance variables, constructors, and methods that are not shown.
}


public class Circle
{
  private Point center;
  private double radius;

  /** Constructs a circle where (a, b) is the center and r is the radius.
   */
  public Circle(double a, double b, double r)
  {
    /* missing code */
  }
}
```

Which of the following replacements for /* missing code */ will correctly implement the Circle constructor?



```
  I.  center = new Point();
      radius = r;

 II.  center = new Point(a, b);
      radius = r;

III.  center = new Point();
      center.x = a;
      center.y = b;
      radius = r;
```

U5-10_2

Ⓐ I only

Ⓑ II only

Ⓒ III only

Ⓓ II and III only

Ⓔ I, II, and III

U5-10_2

12. Consider the following class declarations.

```
public class A
{
private int x;
public A()
{ x = 0; }
public A(int y)
{ x = y; }
// There may be instance variables, constructors, and methods that are not shown.
}

public class B extends A
{
private int y;
public B()
{
/* missing code */
}
// There may be instance variables, constructors, and methods that are not shown.
}
```

Which of the following can be used to replace /* *missing code* */ so that the statement

```
B temp = new B();
```

will construct an object of type B and initialize both x and y with 0 ?

1.
y = 0

2.
super (0);
y = 0;

3.
x = 0;
y = 0;

U5-10_2

Ⓐ I only

Ⓑ II only

Ⓒ I and II only

Ⓓ II and III only

Ⓔ I, II, and III

U5-10_2

13. Consider the following class that stores information about temperature readings on various dates.

```
public class TemperatureReading implements Comparable
{
private double temperature; private int month, day, year;
public int compareTo(Object obj)
{
TemperatureReading other = (TemperatureReading) obj;
/* missing code */
}
// There may be instance variables, constructors, and methods that are not shown.
}
```

Consider the following code segments that are potential replacements for /* *missing code* */.

1.

```
Double d1 = new Double(temperature); Double d2 = new Double(other.temperature);
return d1.compareTo(d2);
```

2.

```
if (temperature < other.temperature)
   return -1;
else if (temperature == other.temperature)
   return 0;
else
   return 1;
```

III. return (int) (temperature - other.temperature);

Which of the code segments could be used to replace /* *missing code* */ so that compareTo can be used to order TemperatureReading objects by increasing temperature value?

(A) II only

(B) I and II only

(C) I and III only

(D) II and III only

(E) I, II, and III

14. Consider the following class.

```
public class SomeMethods
{
public void one(int first)
{ / * implementation not shown * / }
public void one(int first, int second)
{ / * implementation not shown * / }
public void one(int first, String second)
{ / * implementation not shown * / }
}
```

Which of the following methods can be added to the SomeMethods class without causing a compile-time error?

1.

```
public void one(int value)
{ / * implementation not shown * / }
```

2.

```
public void one (String first, int second)
{ / * implementation not shown * / }
```

3.

```
public void one (int first, int second, int third)
{ / * implementation not shown * / }
```

U5-10_2

(A) I only

(B) I and II only

(C) I and III only

(D) II and III only

(E) I, II, and III

15. Consider the following classes.

```
public class Base
{
public Base()
{
System.out.print("Base" + " ");
}
}
public class Derived extends Base
{
public Derived()
{
System.out.print("Derived" + " ");
}
}
```

Assume that the following statement appears in another class.

```
Derived d1 = new Derived();
```

What is printed as a result of executing the statement?

Ⓐ Nothing is printed because the statement is a variable declaration.

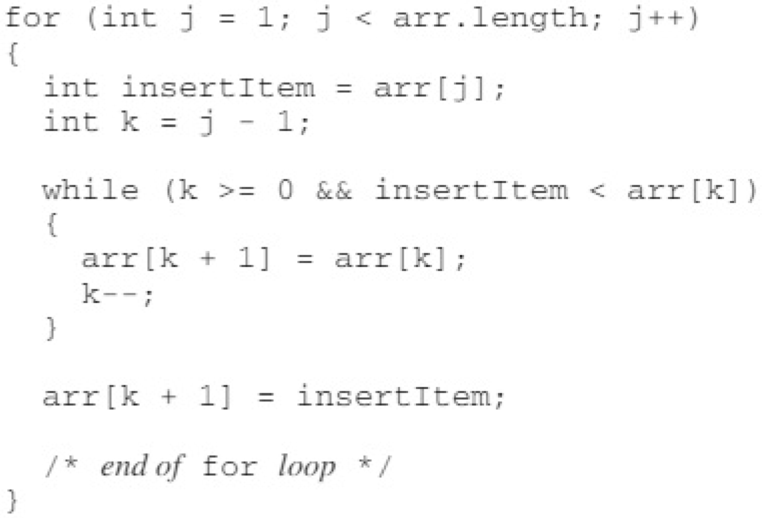
Ⓑ Base

Ⓒ Derived

Ⓓ Base Derived

Ⓔ Derived Base

16. Consider the following code segment from an insertion sort program.



```
for (int j = 1; j < arr.length; j++)
{
  int insertItem = arr[j];
  int k = j - 1;

  while (k >= 0 && insertItem < arr[k])
  {
    arr[k + 1] = arr[k];
    k--;
  }

  arr[k + 1] = insertItem;

  /* end of for loop */
}
```

Assume that array arr has been defined and initialized with the values {5, 4, 3, 2, 1}. What are the values in array arr after two passes of the for loop (i.e., when j = 2 at the point indicated by / * *end of* for *loop* * /) ?

U5-10_2

Ⓐ {2, 3, 4, 5, 1}

Ⓑ {3, 2, 1, 4, 5}

Ⓒ {3, 4, 5, 2, 1}

Ⓓ {3, 5, 2, 3, 1}

Ⓔ {5, 3, 4, 2, 1}

The following questions refer to the following classes:

```
public class First

{

    public String name()

    {

        return "First";

    }

}



public class Second extends First

{

    public void whoRules()

    {
```

U5-10_2

```
        System.out.print(super.name() + " rules");

        System.out.println(" but " + name() + " is even better");

    }

    public String name()

    {

        return "Second";

    }

}




public class Third extends Second

{

    public String name()

    {

        return "Third";

    }

}
```

U5-10_2

17. Consider the following code segment.

```
/* SomeType1 */ varA = new Second();
/* SomeType2 */ varB = new Third();

varA.whoRules();
varB.whoRules();
```

Which of the following could be used to replace /* ***SomeType1*** */ and /* ***SomeType2*** */ so that the code segment will compile without error?

	/* ***SomeType1*** */	/* ***SomeType2*** */
I.	First	Third
II.	Second	Second
III.	Third	Third

Ⓐ I only

Ⓑ II only

Ⓒ III only

Ⓓ I and II

Ⓔ II and III

U5-10_2

18. Consider the following code segment.

```
ArrayList<Integer> nums = new ArrayList<Integer>();
nums.add(new Integer(37));
nums.add(new Integer(3));
nums.add(new Integer(0));
nums.add(1, new Integer(2));
nums.set(0, new Integer(1));
nums.remove(2);
System.out.println(nums);
```

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

Ⓐ [1, 2, 0]

Ⓑ [1, 3, 0]

Ⓒ [1, 3, 2]

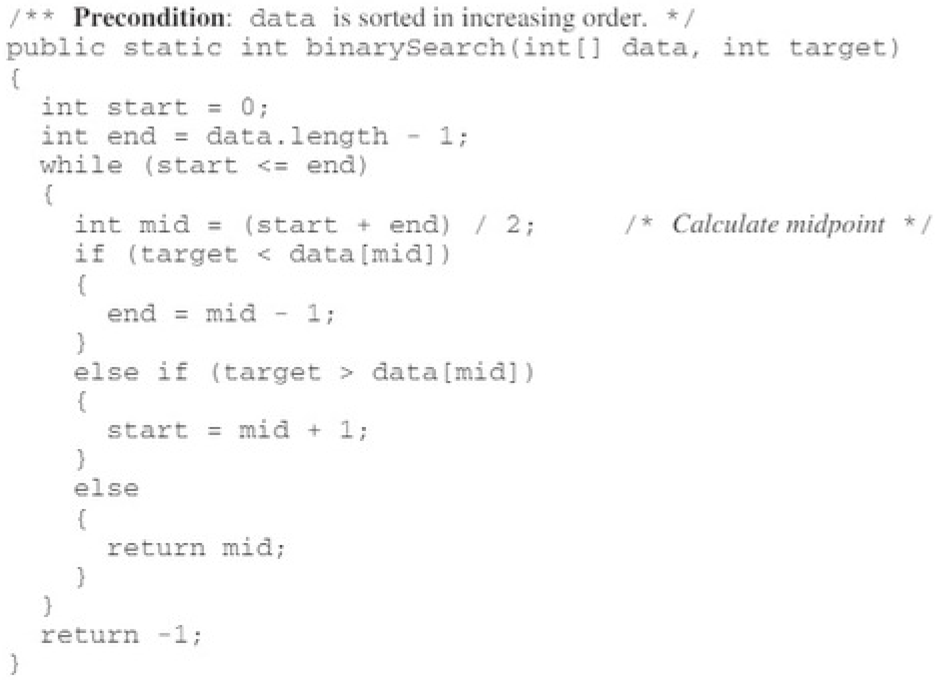
Ⓓ [1, 37, 3, 0]

Ⓔ [37, 0, 0]

Directions: Select the choice that best fits each statement. The following question(s) refer to the following information

Consider the following binarySearch method. The method correctly performs a binary search.

U5-10_2



```
/** Precondition: data is sorted in increasing order. */
public static int binarySearch(int[] data, int target)
{
  int start = 0;
  int end = data.length - 1;
  while (start <= end)
  {
    int mid = (start + end) / 2;      /* Calculate midpoint */
    if (target < data[mid])
    {
      end = mid - 1;
    }
    else if (target > data[mid])
    {
      start = mid + 1;
    }
    else
    {
      return mid;
    }
  }
  return -1;
}
```

19. Consider the following code segment.

int [ ] values = {1, 2, 3, 4, 5, 8, 8, 8};int target = 8;
What value is returned by the call binarySearch (values, target) ?

(A) -1

(B) 3

(C) 5

(D) 6

(E) 8

U5-10_2

20. Consider the following code segment.

```
int[] arr = {4, 2, 9, 7, 3};
for (int k : arr)
{
k = k + 10;
System.out.print(k + " ");
}
for (int k : arr)
System.out.print(k + " ");
```

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

(A) 0 1 2 3 4 0 1 2 3 4

(B) 4 2 9 7 3 4 2 9 7 3

(C) 10 11 12 13 14 0 1 2 3 4

(D) 14 12 19 17 13 4 2 9 7 3

(E) 14 12 19 17 13 14 12 19 17 13