Computer Science A: Sample Multiple-Choice Questions

Following is a representative set of questions. Questions marked with an asterisk are also representative of AB Exam questions. The answer key for the Computer Science A multiple-choice questions is on page 40. In this section of the exam, as a correction for haphazard guessing, one-fourth of the number of questions answered incorrectly will be subtracted from the number of questions answered correctly. The AP Computer Science A Exam will include several multiple-choice questions based on the *AP Computer Science Case Study.* (See AP Central for examples.)

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 found in the Appendix 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 method calls are not `null`.

1. Consider the following code segment.

```
for (int k = 0; k < 20; k = k + 2)
{
  if (k % 3 == 1)
    System.out.print(k + " ");
}
```

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

(A) `4  16`

(B) `4  10  16`

(C) `0  6  12  18`

(D) `1  4  7  10  13  16  19`

(E) `0  2  4  6  8  10  12  14  16  18`

2. Consider the following code segment.

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

list.add("P");
list.add("Q");
list.add("R");
list.set(2, "s");
list.add(2, "T");
list.add("u");
System.out.println(list);
```

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

(A) `[P,  Q,  R,  s,  T]`

(B) `[P,  Q,  s,  T,  u]`

(C) `[P,  Q,  T,  s,  u]`

(D) `[P,  T,  Q,  s,  u]`

(E) `[P,  T,  s,  R,  u]`

*3. Consider the following instance variable and method.

```
private ArrayList<Integer> nums;

/** Precondition: nums.size > 0
 */
public void numQuest()
{
  int k = 0;
  Integer zero = new Integer(0);

  while (k < nums.size())
  {
    if (nums.get(k).equals(zero))
      nums.remove(k);

    k++;
  }
}
```

Assume that `ArrayList nums` initially contains the following `Integer` values.

```
[0, 0, 4, 2, 5, 0, 3, 0]
```

What will `ArrayList nums` contain as a result of executing `numQuest` ?

(A) `[0, 0, 4, 2, 5, 0, 3, 0]`

(B) `[4, 2, 5, 3]`

(C) `[0, 0, 0, 0, 4, 2, 5, 3]`

(D) `[3, 5, 2, 4, 0, 0, 0, 0]`

(E) `[0, 4, 2, 5, 3]`

4. At a certain high school students receive letter grades based on the following scale.

Numeric Score	Letter Grade
93 or above	A
From 84 to 92 inclusive	B
From 75 to 83 inclusive	C
Below 75	F

Which of the following code segments will assign the correct string to `grade` for a given integer score ?

I.
```
if (score >= 93)
  grade = "A";
if (score >= 84 && score <= 92)
  grade = "B";
if (score >= 75 && score <= 83)
  grade = "C";
if (score < 75)
  grade = "F";
```

II.
```
if (score >= 93)
  grade = "A";
if (84 <= score <= 92)
  grade = "B";
if (75 <= score <= 83)
  grade = "C";
if (score < 75)
  grade = "F";
```

III.
```
if (score >= 93)
  grade = "A";
else if (score >= 84)
  grade = "B";
else if (score >= 75)
  grade = "C";
else
  grade = "F";
```

(A) II only

(B) III only

(C) I and II only

(D) I and III only

(E) I, II, and III

5. Consider the following output.

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

Which of the following code segments will produce this output?

(A)

```
for (int j = 1; j <= 5; j++)
{
  for (int k = 1; k <= 5; k++)
  {
    System.out.print(j + " ");
  }
  System.out.println();
}
```

(B)

```
for (int j = 1; j <= 5; j++)
{
  for (int k = 1; k <= j; k++)
  {
    System.out.print(j + " ");
  }
  System.out.println();
}
```

(C)

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

(D)

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

(E)
```
for (int j = 1; j <= 5; j++)
{
  for (int k = j; k <= 5; k++)
  {
    System.out.print(k + " ");
  }
  System.out.println();
}
```

6. A car dealership needs a program to store information about the cars for sale. For each car, they want to keep track of the following information: number of doors (2 or 4), whether the car has air conditioning, and its average number of miles per gallon. Which of the following is the best design?

   (A) Use one class, `Car`, which has three data fields: `int numDoors`, `boolean hasAir`, and `double milesPerGallon`.

   (B) Use four unrelated classes: `Car`, `Doors`, `AirConditioning`, and `MilesPerGallon`.

   (C) Use a class `Car` which has three subclasses: `Doors`, `AirConditioning`, and `MilesPerGallon`.

   (D) Use a class `Car`, which has a subclass `Doors`, with a subclass `AirConditioning`, with a subclass `MilesPerGallon`.

   (E) Use three classes: `Doors`, `AirConditioning`, and `MilesPerGallon`, each with a subclass `Car`.

7. Consider the following declarations.

```
public interface Comparable
{
  int compareTo(Object other);
}
public class SomeClass implements Comparable
{
  // ... other methods not shown
}
```

Which of the following method signatures of `compareTo` will satisfy the `Comparable` interface requirement?

I. `public int compareTo(Object other)`
II. `public int compareTo(SomeClass other)`
III. `public boolean compareTo(Object other)`

(A) I only

(B) II only

(C) III only

(D) I and II only

(E) I, II, and III

Questions 8–9 refer to the following incomplete class declaration.

```
public class TimeRecord
{
  private int hours;
  private int minutes;  // 0<=minutes<60

  public TimeRecord(int h, int m)
  {
    hours = h;
    minutes = m;
  }

  /** @return the number of hours
   */
  public int getHours()
  { /* implementation not shown */ }

  /** @return the number of minutes
   *          Postcondition: 0 ≤ minutes < 60
   */
  public int getMinutes()
  { /* implementation not shown */ }


  /** Adds h hours and m minutes to this TimeRecord.
   *  @param h the number of hours
   *          Precondition: h ≥ 0
   *  @param m the number of minutes
   *          Precondition: m ≥ 0
   */
  public void advance(int h, int m)
  {
    hours = hours + h;
    minutes = minutes + m;

    /* missing code */
  }

  // ... other methods not shown

}
```

8. Which of the following can be used to replace `/* missing code */` so that `advance` will correctly update the time?

(A) `minutes = minutes % 60;`

(B) `minutes = minutes + hours % 60;`

(C)
```
hours = hours + minutes / 60;
minutes = minutes % 60;
```

(D)
```
hours = hours + minutes % 60;
minutes = minutes / 60;
```

(E) `hours = hours + minutes / 60;`

9. Consider the following declaration that appears in a client program.

```
TimeRecord[] timeCards = new TimeRecord[100];
```

Assume that `timeCards` has been initialized with `TimeRecord` objects. Consider the following code segment that is intended to compute the total of all the times stored in `timeCards`.

```
TimeRecord total = new TimeRecord(0,0);

for (int k = 0; k < timeCards.length; k++)
{
  /* missing expression */ ;
}
```

Which of the following can be used to replace `/* missing expression */` so that the code segment will work as intended?

(A) `timeCards[k].advance()`

(B) `total += timeCards[k].advance()`

(C)
```
total.advance(timeCards[k].hours,
              timeCards[k].minutes)
```

(D)
```
total.advance(timeCards[k].getHours(),
              timeCards[k].getMinutes())
```

(E)
```
timeCards[k].advance(timeCards[k].getHours(),
                     timeCards[k].getMinutes())
```

*10. Consider the following instance variable and method.

```
private int[] arr;


/** Precondition: arr contains no duplicates;
 *                the elements in arr are in sorted order.
 *   @param low 0 ≤ low ≤ arr.length
 *   @param high low - 1 ≤ high < arr.length
 *   @param num
 */
public int mystery(int low, int high, int num)
{
  int mid = (low + high) / 2;

  if (low > high)
  {
    return low;
  }
  else if (arr[mid] < num)
  {
    return mystery(mid + 1, high, num);
  }
  else if (arr[mid] > num)
  {
    return mystery(low, mid - 1, num);
  }
  else              // arr[mid] == num
  {
    return mid;
  }
}
```

What is returned by the call
`mystery(0, arr.length - 1, num)` ?

(A) The number of elements in `arr` that are less than `num`

(B) The number of elements in `arr` that are less than or equal to `num`

(C) The number of elements in `arr` that are equal to `num`

(D) The number of elements in `arr` that are greater than `num`

(E) The index of the middle element in `arr`

Questions 11–12 refer to the following information.

Consider the following instance variable and method `findLongest` with line numbers added for reference. Method `findLongest` is intended to find the longest consecutive block of the value `target` occurring in the array `nums;` however, `findLongest` does not work as intended.

For example, if the array `nums` contains the values
[`7, 10, 10, 15, 15, 15, 15, 10, 10, 10, 15, 10, 10`],
the call `findLongest(10)` should return 3, the length of
the longest consecutive block of 10's.

```
    private int[] nums;

    public int findLongest(int target)
    {
      int lenCount = 0;
      int maxLen = 0;

Line 1:  for (val : nums)
Line 2:  {
Line 3:    if (val == target)
Line 4:    {
Line 5:      lenCount++;
Line 6:    }
Line 7:    else
Line 8:    {
Line 9:      if (lenCount > maxLen)
Line 10:     {
Line 11:       maxLen = lenCount;
Line 12:     }
Line 13:   }
Line 14: }
Line 15: if (lenCount > maxLen)
Line 16: {
Line 17:   maxLen = lenCount;
Line 18: }
Line 19: return maxLen;
    }
```

*11. The method `findLongest` does not work as intended.
Which of the following best describes the value returned by a call to `findLongest` ?

(A) It is the length of the shortest consecutive block of the value `target` in `nums`.

(B) It is the length of the array `nums`.

(C) It is the number of occurrences of the value `target` in `nums`.

(D) It is the length of the first consecutive block of the value `target` in `nums`.

(E) It is the length of the last consecutive block of the value `target` in `nums`.

*12. Which of the following changes should be made so that method `findLongest` will work as intended?

(A) Insert the statement `lenCount = 0;` between lines 2 and 3.

(B) Insert the statement `lenCount = 0;` between lines 8 and 9.

(C) Insert the statement `lenCount = 0;` between lines 10 and 11.

(D) Insert the statement `lenCount = 0;` between lines 11 and 12.

(E) Insert the statement `lenCount = 0;` between lines 12 and 13.

*13. Consider the following instance variable and method.

```
private int[] myStuff;

/** Precondition: myStuff contains int values in no particular order.
 */
public int mystery(int num)
{
  for (int k = myStuff.length - 1; k >= 0; k--)
  {
    if (myStuff[k] < num)
    {
      return k;
    }
  }

  return -1;
}
```

Which of the following best describes the contents of `myStuff` after the following statement has been executed?

```
int m = mystery(n);
```

(A) All values in positions `0` through `m` are less than `n`.

(B) All values in positions `m+1` through `myStuff.length-1` are less than `n`.

(C) All values in positions `m+1` through `myStuff.length-1` are greater than or equal to `n`.

(D) The smallest value is at position `m`.

(E) The largest value that is smaller than `n` is at position `m`.

14. Consider the following method.

```
/** @param x x ≥ 0
 */
public void mystery(int x)
{
  System.out.print(x % 10);

  if ((x / 10) != 0)
  {
   mystery(x / 10);
  }

  System.out.print(x % 10);
}
```

Which of the following is printed as a result of the call `mystery(1234)` ?

(A) `1441`

(B) `3443`

(C) `12344321`

(D) `43211234`

(E) Many digits are printed due to infinite recursion.

15. Consider the following two classes.

```
public class Dog
{
  public void act()
  {
    System.out.print("run");
    eat();
  }

  public void eat()
  {
    System.out.print("eat");
  }
}

public class UnderDog extends Dog
{
  public void act()
  {
    super.act();
    System.out.print("sleep");
  }

  public void eat()
  {
    super.eat();
    System.out.print("bark");
  }
}
```

Assume that the following declaration appears in a client program.

```
Dog fido = new UnderDog();
```

What is printed as a result of the call `fido.act()` ?

(A) `run eat`

(B) `run eat sleep`

(C) `run eat sleep bark`

(D) `run eat bark sleep`

(E) Nothing is printed due to infinite recursion.

*16. Consider the following recursive method.

```
public static int mystery(int n)
{
  if (n == 0)
    return 1;
  else
    return 3 * mystery(n - 1);
}
```

What value is returned as a result of the call `mystery(5)` ?

(A) 0

(B) 3

(C) 81

(D) 243

(E) 6561

*17. Consider the following instance variable and method.

```
private int[] arr;

/** Precondition: arr.length > 0
 */
public int checkArray()
{
  int loc = arr.length / 2;

  for (int k = 0; k < arr.length; k++)
  {
    if (arr[k] > arr[loc])
      loc = k;
  }

  return loc;
}
```

Which of the following is the best postcondition for `checkArray` ?

(A) Returns the index of the first element in array `arr` whose value is greater than `arr[loc]`

(B) Returns the index of the last element in array `arr` whose value is greater than `arr[loc]`

(C) Returns the largest value in array `arr`

(D) Returns the index of the largest value in array `arr`

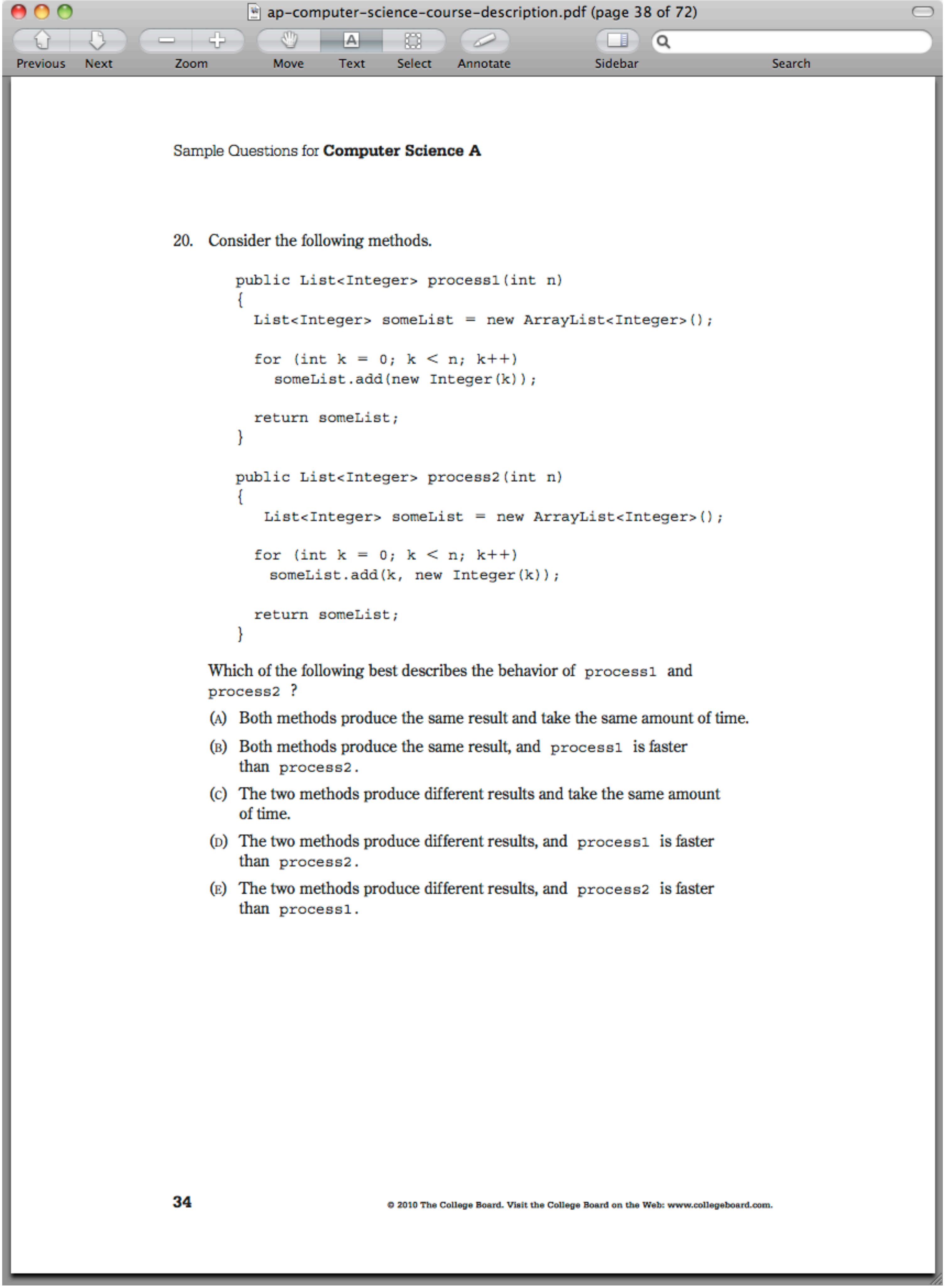
(E) Returns the index of the largest value in the second half of array `arr`

18. Assume the following declarations have been made.

```
private String s;
private int n;

public void changer(String x, int y)
{
  x = x + "peace";
  y = y * 2;
}
```

Assume `s` has the value `"world"` and `n` is `6`. What are the values of `s` and `n` after the call `changer(s, n)` ?

	s	n
(A)	world	6
(B)	worldpeace	6
(C)	world	12
(D)	worldpeace	12
(E)	peace	12



20. Consider the following methods.

```
public List<Integer> process1(int n)
{
  List<Integer> someList = new ArrayList<Integer>();

  for (int k = 0; k < n; k++)
    someList.add(new Integer(k));

  return someList;
}

public List<Integer> process2(int n)
{
   List<Integer> someList = new ArrayList<Integer>();

  for (int k = 0; k < n; k++)
    someList.add(k, new Integer(k));

  return someList;
}
```

Which of the following best describes the behavior of `process1` and `process2` ?

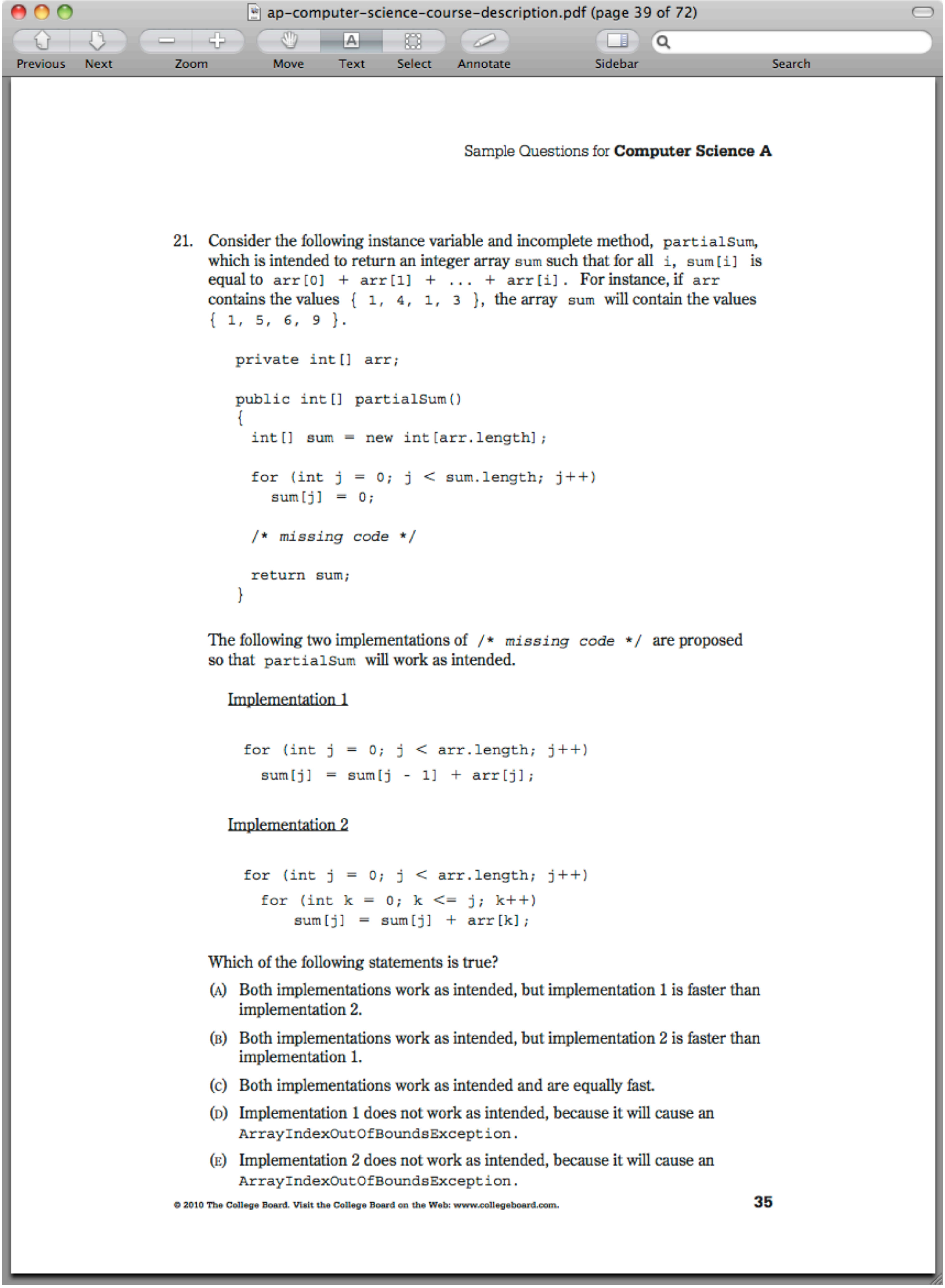
(A) Both methods produce the same result and take the same amount of time.

(B) Both methods produce the same result, and `process1` is faster than `process2`.

(C) The two methods produce different results and take the same amount of time.

(D) The two methods produce different results, and `process1` is faster than `process2`.

(E) The two methods produce different results, and `process2` is faster than `process1`.



21. Consider the following instance variable and incomplete method, `partialSum`, which is intended to return an integer array `sum` such that for all `i`, `sum[i]` is equal to `arr[0] + arr[1] + ... + arr[i]`. For instance, if `arr` contains the values `{ 1, 4, 1, 3 }`, the array `sum` will contain the values `{ 1, 5, 6, 9 }`.

```
private int[] arr;

public int[] partialSum()
{
  int[] sum = new int[arr.length];

  for (int j = 0; j < sum.length; j++)
    sum[j] = 0;

  /* missing code */

  return sum;
}
```

The following two implementations of `/* missing code */` are proposed so that `partialSum` will work as intended.

Implementation 1

```
for (int j = 0; j < arr.length; j++)
  sum[j] = sum[j - 1] + arr[j];
```

Implementation 2

```
for (int j = 0; j < arr.length; j++)
  for (int k = 0; k <= j; k++)
      sum[j] = sum[j] + arr[k];
```

Which of the following statements is true?

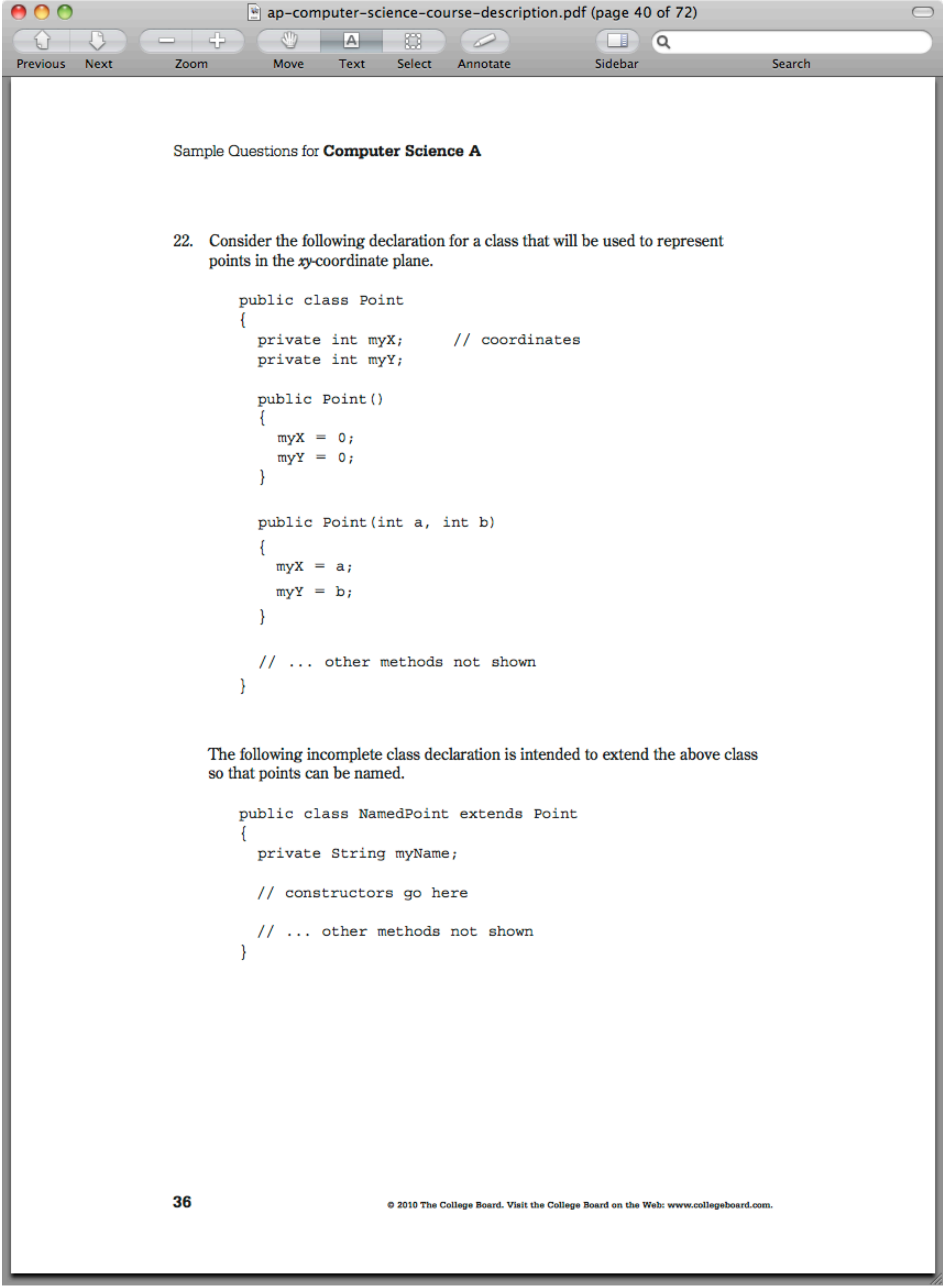
(A) Both implementations work as intended, but implementation 1 is faster than implementation 2.

(B) Both implementations work as intended, but implementation 2 is faster than implementation 1.

(C) Both implementations work as intended and are equally fast.

(D) Implementation 1 does not work as intended, because it will cause an `ArrayIndexOutOfBoundsException`.

(E) Implementation 2 does not work as intended, because it will cause an `ArrayIndexOutOfBoundsException`.

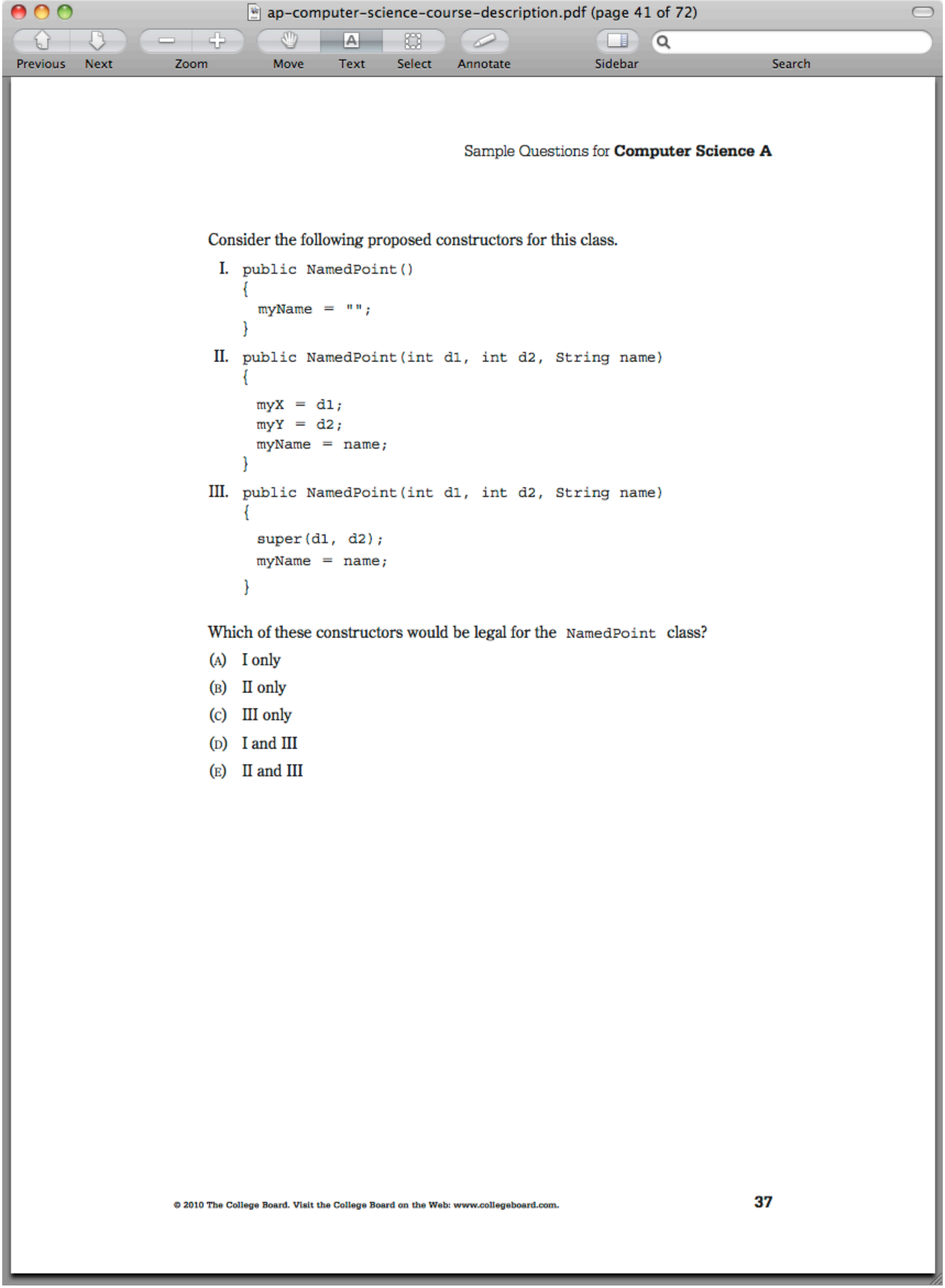


22. Consider the following declaration for a class that will be used to represent points in the *xy*-coordinate plane.

```
public class Point
{
  private int myX;     // coordinates
  private int myY;

  public Point()
  {
    myX = 0;
    myY = 0;
  }

  public Point(int a, int b)
  {
    myX = a;
    myY = b;
  }

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

The following incomplete class declaration is intended to extend the above class so that points can be named.

```
public class NamedPoint extends Point
{
  private String myName;

  // constructors go here

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



Consider the following proposed constructors for this class.

```
  I. public NamedPoint()
     {
       myName = "";
     }
 II. public NamedPoint(int d1, int d2, String name)
     {
       myX = d1;
       myY = d2;
       myName = name;
     }
III. public NamedPoint(int d1, int d2, String name)
     {
       super(d1, d2);
       myName = name;
     }
```

Which of these constructors would be legal for the `NamedPoint` class?

(A) I only

(B) II only

(C) III only

(D) I and III

(E) II and III