# Practice Exam

## Exam Content and Format

The AP Computer Science A Exam is 3 hours long. There are two sections:

- Section I is 1 hour, 30 minutes and consists of 40 multiple-choice questions, accounting for 50 percent of the final score.

- Section II is 1 hour, 30 minutes and consists of 4 free-response questions accounting for 50 percent of the final score.

## Administering the Practice Exam

This section contains instructions for administering the AP Computer Science A Practice Exam. You may wish to use these instructions to create an exam situation that resembles an actual administration. If so, read the indented, boldface directions to the students; all other instructions are for administering the exam and need not be read aloud. Before beginning testing, have all exam materials ready for distribution. These include test booklets and answer sheets. (Reminder: Final instructions for every AP Exam are published in the AP Exam Instructions book.)

### SECTION I: Multiple Choice

When you are ready to begin Section I, say:

> **Section I is the multiple-choice portion of the exam. Mark all of your responses on your answer sheet, one response per question. If you need to erase, do so carefully and completely. Your score on the multiple-choice section will be based solely on the number of questions answered correctly.**
>
> **You have 1 hour and 30 minutes for this part. Open your Section I booklet and begin.**

Note Start Time ________ . Note Stop Time ________ . After 1 hour and 20 minutes, say:

> **There are 10 minutes remaining.**

After 10 minutes, say:

> **Stop working. I will now collect your Section I booklet and multiple-choice answer sheet.**

There is a 10-minute break between Sections I and II.

Name: ______________________________

# AP® Computer Science A Answer Sheet for Multiple-Choice Section

| No. | Answer |
|---|---|
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | |
| 10 | |
| 11 | |
| 12 | |
| 13 | |
| 14 | |
| 15 | |
| 16 | |
| 17 | |
| 18 | |
| 19 | |
| 20 | |

| No. | Answer |
|---|---|
| 21 | |
| 22 | |
| 23 | |
| 24 | |
| 25 | |
| 26 | |
| 27 | |
| 28 | |
| 29 | |
| 30 | |
| 31 | |
| 32 | |
| 33 | |
| 34 | |
| 35 | |
| 36 | |
| 37 | |
| 38 | |
| 39 | |
| 40 | |

# AP® Computer Science A Exam

## SECTION I: Multiple Choice

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

### At a Glance

**Total Time**
1 hour and 30 minutes

**Number of Questions**
40

**Percent of Total Score**
50%

**Writing Instrument**
Pencil required

**Electronic Device**
None allowed

### Instructions

The Java Quick Reference is located inside the front cover of this booklet.

Section I of this exam contains 40 multiple-choice questions.

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.

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 of the multiple-choice questions.

Your total score on the multiple-choice section is based only on the number of questions answered correctly. Points are not deducted for incorrect answers or unanswered questions.

# Java Quick Reference

*Accessible methods from the Java library that may be included in the exam*

| 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()` | |

**COMPUTER SCIENCE A**

**SECTION I**

**Time—1 hour and 30 minutes**

**40 Questions**

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

Notes:

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

**GO ON TO THE NEXT PAGE.**

1. Consider the following code segment.

```
int a = 3 + 2 * 3;
int b = 4 + 3 / 2;
int c = 7 % 4 + 3;
double d = a + b + c;
```

What is the value of `d` after the code segment is executed?

(A) `14.0`

(B) `18.0`

(C) `20.0`

(D) `20.5`

(E) `26.0`

---

2. Consider the following code segment. Assume `num` is a properly declared and initialized `int` variable.

```
if (num > 0)
{
   if (num % 2 == 0)
   {
      System.out.println("A");
   }
   else
   {
      System.out.println("B");
   }
}
```

Which of the following best describes the result of executing the code segment?

(A) When `num` is a negative odd integer, `"B"` is printed; otherwise, `"A"` is printed.

(B) When `num` is a negative even integer, `"B"` is printed; otherwise, nothing is printed.

(C) When `num` is a positive even integer, `"A"` is printed; otherwise, `"B"` is printed.

(D) When `num` is a positive even integer, `"A"` is printed; when `num` is a positive odd integer, `"B"` is printed; otherwise, nothing is printed.

(E) When `num` is a positive odd integer, `"A"` is printed; when `num` is a positive even integer, `"B"` is printed; otherwise, nothing is printed.

**GO ON TO THE NEXT PAGE.**

3. Consider the method `getHours`, which is intended to calculate the number of hours that a vehicle takes to travel between two *mile markers* on a highway if the vehicle travels at a constant speed of 60 miles per hour. A mile marker is a sign showing the number of miles along a road between some fixed location (for example, the beginning of a highway) and the current location.

The following table shows two examples of the intended behavior of `getHours`, based on the `int` parameters `marker1` and `marker2`.

| `marker1` | `marker2` | **Return Value** |
|---|---|---|
| `100` | `220` | `2.0` |
| `100` | `70` | `0.5` |

Consider the following implementation of `getHours`.

```
public static double getHours(int marker1, int marker2)
{
   /* missing statement */
   return hours;
}
```

Which of the following statements can replace `/*` *missing statement* `*/` so `getHours` works as intended?

(A) `double hours = (Math.abs(marker1) - Math.abs(marker2)) / 60.0;`

(B) `double hours = Math.abs(marker1 - marker2 / 60.0);`

(C) `double hours = Math.abs(marker1 - marker2) / 60.0;`

(D) `double hours = Math.abs((marker1 - marker2) / 60);`

(E) `double hours = (double) (Math.abs(marker1 - marker2) / 60);`

**GO ON TO THE NEXT PAGE.**

4. Consider the following method.

```
public static void message(int a, int b, int c)
{
   if (a < 10)
   {
      if (b < 10)
      {
         System.out.print("X");
      }
      System.out.print("Y");
   }
   if (c < 10)
   {
      if (b > 10)
      {
         System.out.print("Y");
      }
      else
      {
         System.out.print("Z");
      }
   }
}
```

What is printed as a result of the call `message(5, 15, 5)` ?

(A) `XY`

(B) `XYZ`

(C) `Y`

(D) `YY`

(E) `Z`

**GO ON TO THE NEXT PAGE.**

5. Consider the following class definition.

```
public class Bird
{
   private String species;
   private String color;
   private boolean canFly;

   public Bird(String str, String col, boolean cf)
   {
      species = str;
      color = col;
      canFly = cf;
   }
}
```

Which of the following constructors, if added to the `Bird` class, will cause a compilation error?

(A)
```
public Bird()
{
   species = "unknown";
   color = "unknown";
   canFly = false;
}
```

(B)
```
public Bird(boolean cf)
{
   species = "unknown";
   color = "unknown";
   canFly = cf;
}
```

(C)
```
public Bird(String col, String str)
{
   species = str;
   color = col;
   canFly = false;
}
```

**GO ON TO THE NEXT PAGE.**

(D)
```
public Bird(boolean cf, String str, String col)
{
   species = str;
   color = col;
   canFly = cf;
}
```

(E)
```
public Bird(String col, String str, boolean cf)
{
   species = str;
   color = col;
   canFly = cf;
}
```

---

6. Which of the following expressions evaluate to `3.5` ?

    I. `(double) 2 / 4 + 3`

    II. `(double) (2 / 4) + 3`

    III. `(double) (2 / 4 + 3)`

(A) I only

(B) III only

(C) I and II only

(D) II and III only

(E) I, II, and III

**GO ON TO THE NEXT PAGE.**

7. Consider the following code segment.

```
int num = /* initial value not shown */;
boolean b1 = true;
if (num > 0)
{
   if (num >= 100)
   {
      b1 = false;
   }
}
else
{
   if (num >= -100)
   {
      b1 = false;
   }
}
```

Which of the following statements assigns the same value to `b2` as the code segment assigns to `b1` for all values of `num` ?

(A) `boolean b2 = (num > -100) && (num < 100);`

(B) `boolean b2 = (num > -100) || (num < 100);`

(C) `boolean b2 = (num < -100) || (num > 100);`

(D) `boolean b2 = (num < -100) && (num > 0 || num < 100);`

(E) `boolean b2 = (num < -100) || (num > 0 && num < 100);`

**GO ON TO THE NEXT PAGE.**

8. Consider the following class definition.

```
public class Points
{
   private double num1;
   private double num2;

   public Points(int n1, int n2)            // Line 6
   {
      num1 = n1;                            // Line 8
      num2 = n2;                            // Line 9
   }

   public void incrementPoints(int value)   // Line 12
   {
      n1 += value;                          // Line 14
      n2 += value;                          // Line 15
   }
}
```

The class does not compile. Which of the following identifies the error in the class definition?

(A) In line 6, the `Points` constructor must have a `void` return type.

(B) In lines 8 and 9, `int` values cannot be assigned to `double` variables.

(C) In line 12, the `incrementPoints` method must have a non-`void` return type.

(D) In lines 14 and 15, the variables `n1` and `n2` are not defined.

(E) In lines 14 and 15, the variable `value` is not defined.

**GO ON TO THE NEXT PAGE.**

9. Consider the following code segment.

```
ArrayList<Integer> numList = new ArrayList<Integer>();

numList.add(3);
numList.add(2);
numList.add(1);
numList.add(1, 0);
numList.set(0, 2);

System.out.print(numList);
```

What is printed by the code segment?

(A) `[1, 3, 0, 1]`

(B) `[2, 0, 2, 1]`

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

(D) `[2, 3, 2, 1]`

(E) `[3, 0, 0, 1]`

10. Consider the following method.

```
public static void printSome(int num1, int num2)
{
   for (int i = 0; i < num1; i++)
   {
      if (i % num2 == 0 && i % 2 == 0)
      {
         System.out.print(i + " ");
      }
   }
}
```

Which of the following method calls will cause `"0 10 "` to be printed?

(A) `printSome(0, 20)`

(B) `printSome(5, 10)`

(C) `printSome(10, 5)`

(D) `printSome(20, 5)`

(E) `printSome(25, 5)`

**GO ON TO THE NEXT PAGE.**

11. Which of the following code segments produces the output `"987654321"` ?

(A)
```
int num = 10;
while (num > 0)
{
   System.out.print(num);
   num--;
}
```

(B)
```
int num = 10;
while (num >= 0)
{
   System.out.print(num);
   num--;
}
```

(C)
```
int num = 10;
while (num > 1)
{
   num--;
   System.out.print(num);
}
```

(D)
```
int num = 10;
while (num >= 1)
{
   num--;
   System.out.print(num);
}
```

(E)
```
int num = 0;
while (num <= 9)
{
   System.out.print(10 - num);
   num++;
}
```

**GO ON TO THE NEXT PAGE.**

12. Consider the following class definitions.

```
public class Person
{
   private String name;

   public String getName()
   {  return name;  }
}

public class Book
{
   private String author;
   private String title;
   private Person borrower;

   public Book(String a, String t)
   {
      author = a;
      title = t;
      borrower = null;
   }

   public void printDetails()
   {
      System.out.print("Author: " + author + " Title: " + title);

      if ( /* missing condition */ )
      {
         System.out.println(" Borrower: " + borrower.getName());
      }
   }

   public void setBorrower(Person b)
   {  borrower = b;  }
}
```

**GO ON TO THE NEXT PAGE.**

Which of the following can replace `/*` *missing condition* `*/` so that the `printDetails` method CANNOT cause a run-time error?

I. `!borrower.equals(null)`

II. `borrower != null`

III. `borrower.getName() != null`

(A) I only

(B) II only

(C) III only

(D) I and II

(E) II and III

---

13. Assume that `a`, `b`, and `c` are `boolean` variables that have been properly declared and initialized. Which of the following `boolean` expressions is equivalent to `!(a && b) || c` ?

(A) `a && b && c`

(B) `a || b || c`

(C) `!a && !b || c`

(D) `!a && !b && c`

(E) `!a || !b || c`

**GO ON TO THE NEXT PAGE.**

14. The following categories are used by some researchers to categorize zip codes as urban, suburban, or rural based on population density.

- An urban zip code is a zip code with more than 3,000 people per square mile.
- A suburban zip code is a zip code with between 1,000 and 3,000 people, inclusive, per square mile.
- A rural zip code is a zip code with fewer than 1,000 people per square mile.

Consider the following method, which is intended to categorize a zip code as urban, suburban, or rural based on the population density of the area included in the zip code.

```
public static String getCategory(int density)
{
   /* missing code */
}
```

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

I.
```
String cat;
if (density > 3000)
{
   cat = "urban";
}
else if (density > 999)
{
   cat = "suburban";
}
else
{
   cat = "rural";
}
return cat;
```

II.
```
String cat;
if (density > 3000)
{
   cat = "urban";
}
if (density > 999)
{
   cat = "suburban";
}
cat = "rural";
return cat;
```

**GO ON TO THE NEXT PAGE.**

III.
```
if (density > 3000)
{
   return "urban";
}
if (density > 999)
{
   return "suburban";
}
return "rural";
```

(A) I only

(B) III only

(C) I and II only

(D) I and III only

(E) I, II, and III

---

15. Consider the following code segment. Assume that `a` is greater than zero.

```
int a = /* value not shown */;
int b = a + (int) (Math.random() * a);
```

Which of the following best describes the value assigned to `b` when the code segment is executed?

(A) `a`

(B) `2 * a`

(C) A random integer between `0` and `a - 1`, inclusive

(D) A random integer between `a` and `2 * a`, inclusive

(E) A random integer between `a` and `2 * a - 1`, inclusive

**GO ON TO THE NEXT PAGE.**

16. Consider the following recursive method.

```
public static void stars(int num)
{
   if (num == 1)
   {
      return;
   }

   stars(num - 1);

   for (int i = 0; i < num; i++)
   {
      System.out.print("*");
   }
   System.out.println();
}
```

What is printed as a result of the method call `stars(5)` ?

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

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

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

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

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

**GO ON TO THE NEXT PAGE.**

17. Consider the following class definitions.

```
public class Hero
{
   private String name;
   private int power;

   public Hero(String n, int p)
   {
      name = n;
      power = p;
   }

   public void powerUp(int p)
   {
      power += p;
   }

   public int showPower()
   {   return power;   }
}

public class SuperHero extends Hero
{
   public SuperHero(String n, int p)
   {
      super(n, p);
   }

   public void powerUp(int p)
   {
      super.powerUp(p * 2);
   }
}
```

The following code segment appears in a class other than `Hero` and `SuperHero`.

```
Hero j = new SuperHero("JavaHero", 50);
j.powerUp(10);
System.out.println(j.showPower());
```

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

(A) `10`

(B) `20`

(C) `60`

(D) `70`

(E) `100`

**GO ON TO THE NEXT PAGE.**

18. Consider the following method, which is intended to return the number of *local maximum* values in an array. Local maximum values are array elements that are greater than both adjacent array elements. The first and last elements of an array have only a single adjacent element, so neither the first nor the last array element is counted by this method. For example, an array containing the values `{3, 9, 7, 4, 10, 12, 3, 8}` has two local maximum values: `9` and `12`.

```
public static int countPeaks(int[] data)
{
   int numPeaks = 0;

   for ( /* missing loop header */ )
   {
      if (data[p - 1] < data[p] && data[p] > data[p + 1])
      {
         numPeaks++;
      }
   }
   return numPeaks;
}
```

Which of the following can replace `/*` *missing loop header* `*/` so the method `countPeaks` works as intended?

(A) `int p = data.length - 1; p > 0; p--`

(B) `int p = 0; p < data.length; p++`

(C) `int p = 0; p < data.length - 1; p++`

(D) `int p = 1; p < data.length; p++`

(E) `int p = 1; p < data.length - 1; p++`

**GO ON TO THE NEXT PAGE.**

19. Consider the following code segment.

```
int[][] values = {{1, 2, 3}, {4, 5, 6}};
int x = 0;

for (int j = 0; j < values.length; j++)
{
   for (int k = 0; k < values[0].length; k++)
   {
      if (k == 0)
      {
         values[j][k] *= 2;
      }
      x += values[j][k];
   }
}
```

What is the value of `x` after the code segment is executed?

(A)  7

(B) 17

(C) 21

(D) 26

(E) 27

GO ON TO THE NEXT PAGE.

20. Consider the following class definition.

```
public class Book
{
   private int pages;

   public int getPages()
   {
      return pages;
   }

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

The following code segment is intended to store in `maxPages` the greatest number of pages found in any `Book` object in the array `bookArr`.

```
Book[] bookArr = { /* initial values not shown */ };
int maxPages = bookArr[0].getPages();

for (Book b : bookArr)
{
   /* missing code */
}
```

**GO ON TO THE NEXT PAGE.**

Which of the following can replace `/*` *missing code* `*/` so the code segment works as intended?

(A)
```
if (b.pages > maxPages)
{
   maxPages = b.pages;
}
```

(B)
```
if (b.getPages() > maxPages)
{
   maxPages = b.getPages();
}
```

(C)
```
if (Book[b].pages > maxPages)
{
   maxPages = Book[b].pages;
}
```

(D)
```
if (bookArr[b].pages > maxPages)
{
   maxPages = bookArr[b].pages;
}
```

(E)
```
if (bookArr[b].getPages() > maxPages)
{
   maxPages = bookArr[b].getPages();
}
```

**GO ON TO THE NEXT PAGE.**

**Questions 21 - 22 refer to the information below.**

Consider the following method.

```
public static String[] strArrMethod(String[] arr)
{
   String[] result = new String[arr.length];

   for (int j = 0; j < arr.length; j++)
   {
      String sm = arr[j];
      for (int k = j + 1; k < arr.length; k++)
      {
         if (arr[k].length() < sm.length())
         {
            sm = arr[k];    // Line 12
         }
      }
      result[j] = sm;
   }
   return result;
}
```

21. Consider the following code segment.

```
String[] testOne = {"first", "day", "of", "spring"};
String[] resultOne = strArrMethod(testOne);
```

What are the contents of `resultOne` when the code segment has been executed?

(A) `{"day", "first", "of", "spring"}`

(B) `{"of", "day", "first", "spring"}`

(C) `{"of", "day", "of", "spring"}`

(D) `{"of", "of", "of", "spring"}`

(E) `{"spring", "first", "day", "of"}`

**GO ON TO THE NEXT PAGE.**

22. Consider the following code segment.

```
String[] testTwo = {"last", "day", "of", "the", "school", "year"};
String[] resultTwo = strArrMethod(testTwo);
```

How many times is the line labeled `//` Line 12 in the `strArrMethod` executed as a result of executing the code segment?

(A) 4 times

(B) 5 times

(C) 6 times

(D) 15 times

(E) 30 times

**GO ON TO THE NEXT PAGE.**

23. Consider the following method, which is intended to print the values in its two-dimensional integer array parameter in row-major order.

```
public static void rowMajor(int[][] arr)
{
   /* missing code */
}
```

As an example, consider the following code segment.

```
int[][] theArray = {{1, 2}, {3, 4}, {5, 6}, {7, 8}};
rowMajor(theArray);
```

When executed, the code segment should produce the following output.

```
1 2 3 4 5 6 7 8
```

**GO ON TO THE NEXT PAGE.**

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

(A)
```
for (int j : arr)
{
   for (int k : j)
   {
      System.out.print(j + " ");
   }
}
```

(B)
```
for (int j : arr)
{
   for (int k : j)
   {
      System.out.print(k + " ");
   }
}
```

(C)
```
for (int[] j : arr)
{
   for (int k : j)
   {
      System.out.print(j + " ");
   }
}
```

(D)
```
for (int[] j : arr)
{
   for (int k : j)
   {
      System.out.print(k + " ");
   }
}
```

(E)
```
for (int[] j : arr)
{
   for (int k : j)
   {
      System.out.print(arr[k] + " ");
   }
}
```

**GO ON TO THE NEXT PAGE.**

24. Consider the following class definition.

```
public class SomeClass
{
   private int x = 0;
   private static int y = 0;

   public SomeClass(int pX)
   {
      x = pX;
      y++;
   }

   public void incrementY()
   {   y++;  }

   public void incrementY(int inc)
   {   y += inc;  }

   public int getY()
   {   return y;  }
}
```

The following code segment appears in a class other than `SomeClass`.

```
SomeClass first = new SomeClass(10);
SomeClass second = new SomeClass(20);
SomeClass third = new SomeClass(30);
first.incrementY();
second.incrementY(10);
System.out.println(third.getY());
```

What is printed as a result of executing the code segment if the code segment is the first use of a `SomeClass` object?

(A) `0`

(B) `1`

(C) `11`

(D) `14`

(E) `30`

**GO ON TO THE NEXT PAGE.**

25. Consider the following method.

```
public static String rearrange(String str)
{
   String temp = "";

   for (int i = str.length() - 1; i > 0; i--)
   {
      temp += str.substring(i - 1, i);
   }

   return temp;
}
```

What, if anything, is returned by the method call `rearrange("apple")` ?

(A) `"appl"`

(B) `"apple"`

(C) `"elppa"`

(D) `"lppa"`

(E) Nothing is returned due to a run-time error.

**GO ON TO THE NEXT PAGE.**

26. Consider the following two code segments. Assume that the `int` variables `m` and `n` have been properly declared and initialized and are both greater than `0`.

I.
```
for (int i = 0; i < m * n; i++)
{
   System.out.print("A");
}
```

II.
```
for (int j = 1; j <= m; j++)
{
   for (int k = 1; k < n; k++)
   {
      System.out.print("B");
   }
}
```

Assume that the initial values of `m` and `n` are the same in code segment I as they are in code segment II. Which of the following correctly compares the number of times that `"A"` and `"B"` are printed when each code segment is executed?

(A) `"A"` is printed `m` fewer times than `"B"`.

(B) `"A"` is printed `n` fewer times than `"B"`.

(C) `"A"` is printed `m` more times than `"B"`.

(D) `"A"` is printed `n` more times than `"B"`.

(E) `"A"` and `"B"` are printed the same number of times.

**GO ON TO THE NEXT PAGE.**

27. Consider the following statement. Assume that `a` and `b` are properly declared and initialized `boolean` variables.

```
boolean c = (a && b) || (!a && b);
```

Under which of the following conditions will `c` be assigned the value `false` ?

(A) Always

(B) Never

(C) When `a` and `b` have the same value

(D) When `a` has the value `false`

(E) When `b` has the value `false`

---

28. Consider the following method.

```
public static String abMethod(String a, String b)
{
   int x = a.indexOf(b);

   while (x >= 0)
   {
      a = a.substring(0, x) + a.substring(x + b.length());
      x = a.indexOf(b);
   }

   return a;
}
```

What, if anything, is returned by the method call `abMethod("sing the song", "ng")` ?

(A) `"si"`

(B) `"si the so"`

(C) `"si the song"`

(D) `"sig the sog"`

(E) Nothing is returned because a `StringIndexOutOfBoundsException` is thrown.

**GO ON TO THE NEXT PAGE.**

29. Consider the following method.

```
public static int calcMethod(int num)
{
   if (num == 0)
   {
      return 10;
   }
   return num + calcMethod(num / 2);
}
```

What value is returned by the method call `calcMethod(16)` ?

(A) `10`

(B) `26`

(C) `31`

(D) `38`

(E) `41`

**GO ON TO THE NEXT PAGE.**

30. Consider the following class definitions.

```
public class Rectangle
{
   private int height;
   private int width;

   public Rectangle()
   {
      height = 1;
      width = 1;
   }

   public Rectangle(int x)
   {
      height = x;
      width = x;
   }

   public Rectangle(int h, int w)
   {
      height = h;
      width = w;
   }

   // There may be methods that are not shown.
}

public class Square extends Rectangle
{
   public Square(int x)
   {
      /* missing code */
   }
}
```

Which of the following code segments can replace `/*` *missing code* `*/` so that the `Square` class constructor initializes the `Rectangle` class instance variables `height` and `width` to `x` ?

(A) `super();`

(B) `super(x);`

(C) `Rectangle(x);`

(D) `Square(x, x);`

(E)
```
height = x;
width = x;
```

**GO ON TO THE NEXT PAGE.**

31. Consider an integer array `nums`, which has been properly declared and initialized with one or more values. Which of the following code segments counts the number of negative values found in `nums` and stores the count in `counter` ?

I.
```
int counter = 0;
int i = -1;
while (i <= nums.length - 2)
{
   i++;
   if (nums[i] < 0)
   {
      counter++;
   }
}
```

II.
```
int counter = 0;
for (int i = 1; i < nums.length; i++)
{
   if (nums[i] < 0)
   {
      counter++;
   }
}
```

III.
```
int counter = 0;
for (int i : nums)
{
   if (nums[i] < 0)
   {
      counter++;
   }
}
```

(A) I only

(B) II only

(C) I and II only

(D) I and III only

(E) I, II, and III

GO ON TO THE NEXT PAGE.

32. Consider the following class definitions.

```
public class ClassA
{
   public String getValue()
   {
      return "A";
   }

   public void showValue()
   {
      System.out.print(getValue());
   }
}

public class ClassB extends ClassA
{
   public String getValue()
   {
      return "B";
   }
}
```

The following code segment appears in a class other than `ClassA` or `ClassB`.

```
ClassA obj = new ClassB();
obj.showValue();
```

What, if anything, is printed when the code segment is executed?

(A) `A`

(B) `B`

(C) `AB`

(D) `BA`

(E) Nothing is printed because the code does not compile.

**GO ON TO THE NEXT PAGE.**

33. Consider the following code segment.

```
String[][] letters = {{"A", "B", "C", "D"},
                      {"E", "F", "G", "H"},
                      {"I", "J", "K", "L"}};

for (int col = 1; col < letters[0].length; col++)
{
   for (int row = 1; row < letters.length; row++)
   {
      System.out.print(letters[row][col] + " ");
   }

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

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

(A)
```
A E I
F J
K
```

(B)
```
B F J
C G K
D H L
```

(C)
```
E I
F J
G K
H L
```

(D)
```
F G H
J K L
```

(E)
```
F J
G K
H L
```

**GO ON TO THE NEXT PAGE.**

34. The following method is intended to remove all elements of an `ArrayList` of integers that are divisible by `key` and add the removed elements to a new `ArrayList`, which the method returns.

```
public static ArrayList<Integer> match(ArrayList<Integer> numList, int key)
{
   ArrayList<Integer> returnList = new ArrayList<Integer>();

   int i = 0;
   while (i < numList.size())
   {
      int num = numList.get(i);
      if (num % key == 0)
      {
         numList.remove(i);
         returnList.add(num);
      }
      i++;
   }
   return returnList;
}
```

As an example, if the method is called with an `ArrayList` containing the values `[5, 2, 10, 20, 16]` and the parameter `key` has the value `5`, then `numList` should contain `[2, 16]` at the end of the method and an `ArrayList` containing `[5, 10, 20]` should be returned.

Which of the following best explains why the method does not always work as intended?

(A) The method attempts to add an element to `returnList` after that element has already been removed from `numList`.

(B) The method causes a `NullPointerException` to be thrown when no matches are found.

(C) The method causes an `IndexOutOfBoundsException` to be thrown.

(D) The method fails to correctly determine whether an element of `numList` is divisible by `key`.

(E) The method skips some elements of `numList` during the traversal.

**GO ON TO THE NEXT PAGE.**

35. Consider the `mode` method, which is intended to return the most frequently occurring value (mode) in its `int[]` parameter `arr`. For example, if the parameter of the `mode` method has the contents `{6, 5, 1, 5, 2, 6, 5}`, then the method is intended to return `5`.

```
/** Precondition: arr.length >= 1 */
public static int mode(int[] arr)
{
   int modeCount = 1;
   int mode = arr[0];

   for (int j = 0; j < arr.length; j++)
   {
      int valCount = 0;
      for (int k = 0; k < arr.length; k++)
      {
         if ( /* missing condition 1 */ )
         {
            valCount++;
         }
      }
      if ( /* missing condition 2 */ )
      {
         modeCount = valCount;
         mode = arr[j];
      }
   }
   return mode;
}
```

Which of the following can replace `/* missing condition 1 */` and `/* missing condition 2 */` so the code segment works as intended?

| | `/* missing condition 1 */` | `/* missing condition 2 */` |
|---|---|---|
| (A) | `arr[j] == arr[k]` | `valCount > modeCount` |
| (B) | `arr[j] == arr[k]` | `modeCount > valCount` |
| (C) | `arr[j] != arr[k]` | `valCount > modeCount` |
| (D) | `arr[j] != arr[k]` | `modeCount > valCount` |
| (E) | `arr[j] != arr[k]` | `modeCount != valCount` |

**GO ON TO THE NEXT PAGE.**

36. Consider the following methods.

```
/** Precondition: a > 0 and b > 0 */
public static int methodOne(int a, int b)
{
   int loopCount = 0;
   for (int i = 0; i < a / b; i++)
   {
      loopCount++;
   }
   return loopCount;
}

/** Precondition: a > 0 and b > 0 */
public static int methodTwo(int a, int b)
{
   int loopCount = 0;
   int i = 0;
   while (i < a)
   {
      loopCount++;
      i += b;
   }
   return loopCount;
}
```

Which of the following best describes the conditions under which `methodOne` and `methodTwo` return the same value?

(A) When `a` and `b` are both even

(B) When `a` and `b` are both odd

(C) When `a` is even and `b` is odd

(D) When `a % b` is equal to zero

(E) When `a % b` is equal to one

**GO ON TO THE NEXT PAGE.**

37. Consider the following code segment. Assume that `num3 > num2 > 0`.

```
int num1 = 0;
int num2 = /* initial value not shown */;
int num3 = /* initial value not shown */;

while (num2 < num3)
{
   num1 += num2;
   num2++;
}
```

Which of the following best describes the contents of `num1` as a result of executing the code segment?

(A) The product of `num2` and `num3`

(B) The product of `num2` and `num3 - 1`

(C) The sum of `num2` and `num3`

(D) The sum of all integers from `num2` to `num3`, inclusive

(E) The sum of all integers from `num2` to `num3 - 1`, inclusive

**GO ON TO THE NEXT PAGE.**

38. Consider the following class definition.

```
public class Value
{
   private int num;

   public int getNum()
   {
      return num;
   }

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

The following method appears in a class other than `Value`. It is intended to sum all the `num` instance variables of the `Value` objects in its `ArrayList` parameter.

```
/** Precondition: valueList is not null */
public static int getTotal(ArrayList<Value> valueList)
{
   int total = 0;
   /* missing code */
   return total;
}
```

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

```
  I. for (int x = 0; x < valueList.size(); x++)
     {
        total += valueList.get(x).getNum();
     }

 II. for (Value v : valueList)
     {
        total += v.getNum();
     }

III. for (Value v : valueList)
     {
        total += getNum(v);
     }
```

(A) I only

(B) II only

(C) III only

(D) I and II

(E) I and III

**GO ON TO THE NEXT PAGE.**

39. Consider the following recursive method.

```
public static boolean recurMethod(String str)
{
   if (str.length() <= 1)
   {
      return true;
   }
   else if (str.substring(0, 1).compareTo(str.substring(1, 2)) > 0)
   {
      return recurMethod(str.substring(1));
   }
   else
   {
      return false;
   }
}
```

Which of the following method calls will return `true` ?

(A) `recurMethod("abcba")`

(B) `recurMethod("abcde")`

(C) `recurMethod("bcdab")`

(D) `recurMethod("edcba")`

(E) `recurMethod("edcde")`

**GO ON TO THE NEXT PAGE.**

40. Consider the following class definitions.

```
public class A
{
   public String message(int i)
   {
      return "A" + i;
   }
}

public class B extends A
{
   public String message(int i)
   {
      return "B" + i;
   }
}
```

The following code segment appears in a class other than `A` or `B`.

```
A obj1 = new B();                          // Line 1
B obj2 = new B();                          // Line 2
System.out.println(obj1.message(3));   // Line 3
System.out.println(obj2.message(2));   // Line 4
```

Which of the following best explains the difference, if any, in the behavior of the code segment that will result from removing the `message` method from class `A` ?

(A) The statement in line 3 will cause a compiler error because the `message` method for `obj1` cannot be found.

(B) The statement in line 4 will cause a compiler error because the `message` method for `obj2` cannot be found.

(C) As a result of the method call in line 3, the `message` method in class `B` will be executed instead of the `message` method in class `A`.

(D) As a result of the method call in line 4, the `message` method in class `B` will be executed instead of the `message` method in class `A`.

(E) The behavior of the code segment will remain unchanged.