AP Computer Science A: Practice Exam 2

Part I (Multiple Choice)

Time: 90 minutes
Number of questions: 40
Percent of total score: 50

Directions: Choose the best answer for each problem. Some problems take longer than others. Consider how much time you have left before spending too much time on any one problem.

Notes:

- You may assume all import statements have been included where they are needed.
- You may assume that the parameters in method calls are not null.
- You may assume that declarations of variables and methods appear within the context of an enclosing class.

1. Consider the following code segment.

```
for (int h = 5; h >= 0; h = h - 2)
{
   for (int k = 0; k < 3; k++)
   {
      if ((h + k) % 2 == 0)
         System.out.print((h + k) + "  ");
   }
}
```

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

(A) `5  3  1`
(B) `6  4  2`
(C) `51  31  11`
(D) `6  4  2  0`
(E) `51  31  11  00`

2. Consider the following code segment.

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

foods.add("Hummus");
foods.add("Soup");
foods.add("Sushi");
foods.set(1, "Empanadas");
foods.add(0, "Salad");
foods.remove(1);
foods.add("Curry");
System.out.println(foods);
```

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

(A) `[Salad, Empanadas, Sushi, Curry]`
(B) `[Hummus, Salad, Empanadas, Sushi, Curry]`
(C) `[Hummus, Soup, Sushi, Empanadas, Salad, Curry]`
(D) `[Soup, Empanadas, Sushi, Curry]`
(E) `[Hummus, Sushi, Salad, Curry]`

3. Consider the following output.

```
A  A  A  A  P  P  P  P  C  C  C  C
```

Which of the following code segments will produce this output?

I.
```
String[] letters = {"A", "P", "C", "S"};
for (int i = letters.length; i > 0; i--)
{
   if (!letters[i].equals("S"))
   {
      for (int k = 0; k < 4; k++)
         System.out.print(letters[k] + "  ");
   }
}
```

II.
```
String letters = "APCS";
for (int i = 0; i < letters.length(); i++)
{
   String s = letters.substring(i, i + 1);
   for (int num = 0; num < 4; num++)
   {
      if (!s.equals("S"))
         System.out.print(s + "  ");
   }
}
```

III.
```
String[][] letters = { {"A", "P"}, {"C", "S"} };
for (String[] row : letters)
{
   for (String letter : row)
   {
      if (!letter.equals("S"))
         System.out.print(letter + "  ");
   }
}
```

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

Questions 4–5 refer to the following two classes.

```
public class Vehicle
{
   private int fuel;

   public Vehicle(int fuelAmt)
   {
      fuel = fuelAmt;
   }

   public void start()
   {
      System.out.println("Vroom");
   }

   public void changeFuel(int change)
   {
      fuel = fuel + change;
   }

   public int getFuel()
   {
      return fuel;
   }
}
public class Boat extends Vehicle
{
   public Boat(int fuelAmount)
   {
       super(fuelAmount);
   }

   public void useFuel()
   {
      super.changeFuel(-2);
   }

   public void start()
   {
      super.start();
      useFuel();
      System.out.println("Remaining Fuel " + getFuel());
   }
}
```

4. Assume the following declaration appears in a client program.

```
Boat yacht = new Boat(20);
```

What is printed as a result of executing the call `yacht.start()`?

(A) `Vroom`
(B) `Remaining Fuel 18`
(C) `Remaining Fuel 20`
(D) `Vroom`
`Remaining Fuel 18`
(E) `Vroom`
`Remaining Fuel 20`

5. Which of the following statements results in a compile-time error?

I. `Boat sailboat = new Boat(2);`
II. `Vehicle tanker = new Boat(20);`
III. `Boat tugboat = new Vehicle(10);`

(A) I only
(B) II only
(C) III only
(D) II and III only
(E) None of these options will result in a compile-time error.

6. Consider the following recursive method.

```
public int weird(int num)
{
   if (num <= 0)
      return num;
   return weird(num - 2) + weird(num - 1) + weird(num);
}
```

What value is returned as a result of the call `weird(3)`?

(A) `-2`
(B) `3`
(C) `4`
(D) `6`
(E) Nothing is returned. Infinite recursion causes a stack overflow error.

7. Consider the following method.

```
public boolean verifyValues(int[] values)
{
   for (int index = values.length - 1; index > 0; index--)
   {
      /* missing code */
         return false;
   }
   return true;
}
```

The method `verifyValues` is intended to return true if the array passed as a parameter is in ascending order (least to greatest), and `false` otherwise.

Which of the following lines of code could replace `/* missing code */` so the method works as intended?

(A) `if (values[index] <= values[index - 1])`
(B) `if (values[index + 1] < values[index])`
(C) `if (values[index] >= values[index - 1])`
(D) `if (values[index - 1] > values[index])`
(E) `if (values[index] < values[index + 1])`

8. Consider the following code segment.

```
int[][] matrix = new int[3][3];
int value = 0;

for (int row = 0; row < matrix.length; row++)
{
   for (int column = 0; column < matrix[row].length; column++)
   {
      matrix[row][column] = value;
      value++;
   }
}

int sum = 0;
for (int[] row : matrix)
{
   for (int number : row)
      sum = sum + number;
}
```

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

(A) `0`
(B) `9`
(C) `21`
(D) `36`
(E) `72`

9. Consider the following class used to represent a student.

```
public class Student
{
   private String name;
   private int year;

   public Student()
   {
      name = "name";
      year = 0;
   }

   public Student(String myName, int myYear)
   {
      name = myName;
      year = myYear;
   }
}
```

Consider the `ExchangeStudent` class that extends the `Student` class.

```
public class ExchangeStudent extends Student
{
   private String country;
   private String language;

   public ExchangeStudent(String myName, int myYear,
                          String myCountry, String myLanguage)
   {
      super(myName, myYear);
      country = myCountry;
      language = myLanguage;
   }
}
```

Which of the following constructors could also be included in the `ExchangeStudent` class without generating a compile-time error?

I.
```
public ExchangeStudent(String myName, int myYear, String myCountry)
{
   super(myName, myYear);
   country = myCountry;
   language = "English";
}
```

II.
```
public ExchangeStudent(String myCountry, String myLanguage)
{
   super("name", "2015");
   country = myCountry;
   language = myLanguage;
}
```

III.
```
public ExchangeStudent()
{
}
```

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

10. Consider the following code segment.

```
int number = Integer.MAX_VALUE;
while (number > 0)
{
   number = number / 2;
}

for (int i = number; i > 0; i++)
{
   System.out.print(i + "  ");
}
```

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

(A) `0`
(B) `1`
(C) `1073741823`
(D) Nothing will be printed. The code segment will terminate without error.
(E) Nothing will be printed. The first loop is an infinite loop.

11. Consider the following method.

```
/** Precondition: numbers.size() > 0
 */
public int totalValue(ArrayList<Integer> numbers)
{
   int total = 0;
   for (Integer val : numbers)
   {
      if (val > 1 && numbers.size() - 3 > val)
         total += val;
   }
   return total;
}
```

Assume that the `ArrayList` passed as a parameter contains the following `Integer` values.

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

What value is returned by the call `totalValue`?

(A) `20`
(B) `21`
(C) `27`
(D) `44`
(E) `45`

12. Given the String declaration

```
String course = "AP Computer Science A";
```

What value is returned by the call course.indexOf("e")?

(A) 9
(B) 10
(C) 18
(D) 9, 15, 18
(E) 10, 16, 19

13. Consider the code segment.

```
if (value == 1)
   count += 1;
else if (value == 2)
   count += 2;
```

Which segment could be used to replace the segment above and work as intended?

(A)
```
if (value == 1 || value == 2)
        count += 1;
```
(B)
```
if (value == 1 || value == 2)
        count += 2;
```
(C)
```
if (value == 1 || value == 2)
        count += value;
```
(D)
```
if (value == 1 && value == 2)
        count += 1;
```
(E)
```
if (value == 1 && value == 2)
        count += value;
```

14. A bank will approve a loan for any customer who fulfills one or more of the following requirements:

- Has a credit score ≥ 640
- Has a cosigner for the loan
- Has a credit score ≥ 590 and has collateral

Which of the following methods will properly evaluate the customer's eligibility for a loan?

I.
```
public boolean canGetLoan(int credit, boolean cosigner, boolean coll)
{
   if (credit >= 640 || cosigner || (credit >= 590 && coll))
      return true;
   return false;
}
```

II.
```
public boolean canGetloan(int credit, boolean cosigner, boolean coll)
{
   if (!credit >= 640 || !cosigner || !(credit >= 590 && coll))
      return false;
   else
      return true;
}
```

III.
```
public boolean canGetLoan(int credit, boolean cosigner, boolean coll)
{
   if (coll && credit >= 590)
      return true;
   if (credit >= 640)
      return true;
   if (cosigner)
      return true;
   return false;
}
```

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

15. Consider the following code segment.

```
int value = 33;
boolean calculate = true;
while (value > 5 || calculate)
{
   if (value % 3 == 0)
      value = value - 2;
   if (value / 4 < 3)
      calculate = false;
}
System.out.print(value);
```

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

(A) `0`
(B) `1`
(C) `8`
(D) `33`
(E) Nothing will be printed. It is an infinite loop.

GO ON TO THE NEXT PAGE

16. Assume that `planets` has been correctly instantiated and initialized to contain the names of the planets. Which of the following code segments will reverse the order of the elements in `planets`?

(A)
```
for (int index = planets.length / 2; index >= 0; index--)
{
   String temp = planets[index];
   planets[index] = planets[index + 1];
   planets[index + 1] = temp;
}
```

(B)
```
int index = planets.length / 2;
while (index >= 0)
{
   String s = planets[index];
   planets[index] = planets[index + 1];
   index--;
}
```

(C)
```
String[] newPlanets = new String[planets.length];
for (int i = planets.length - 1; i >= 0; i--)
{
   newPlanets[i] = planets[i];
}
planets = newPlanets;
```

(D)
```
for (int index = 0; index < planets.length; index++)
{
   String temp = planets[index];
   planets[index] = planets[planets.length - 1 - index];
   planets[planets.length - 1 - index] = temp;
}
```

(E)
```
for (int index = 0; index < planets.length / 2; index++)
{
   String temp = planets[index];
   planets[index] = planets[planets.length - 1 - index];
   planets[planets.length - 1 - index] = temp;
}
```

17. Consider the program segment.

```
for (int i = start; i <= stop; i++)
    System.out.println("AP Computer Science A Rocks!!!");
```

How many times will "AP Computer Science A Rocks!!!" be printed?

(A) `stop - start - 1`
(B) `stop - start`
(C) `stop - start + 1`
(D) `stop -1`
(E) `stop`

18. Consider the following code segment.

```
int num = (int)(Math.random() * 30 + 20);
num += 5;
num = num / 5;
System.out.print(num);
```

What are the possible values that could be printed to the console?

(A) All real numbers from 4 to 10 (not including 10)
(B) All integers from 5 to 10 (inclusive)
(C) All integers from 20 to 49 (inclusive)
(D) All integers from 25 to 54 (inclusive)
(E) All real numbers from 30 to 50 (inclusive)

19. Consider the following code segment.

```
String exampleString = "computer";
ArrayList<String> words = new ArrayList<String>();
for (int k = 0; k < exampleString.length(); k++)
{
   words.add(exampleString.substring(k));
   k++;
}

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

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

(A) `[computer, mputer, uter, er]`
(B) `[computer, comput, comp, co]`
(C) `[computer, computer, computer, computer]`
(D) `[computer, omputer, mputer, puter, uter, ter, er, r]`
(E) Nothing is printed. There is an `ArrayListIndexOutOfBoundsException`.

20. Consider the following incomplete method.

```
public boolean validation(int x, int y)
{
   /* missing code */
}
```

The following table shows several examples of the desired result of a call to `validation`.

x	y	Result
1	0	true
3	6	true
1	3	false
2	6	false

Which of the following code segments should replace `/* missing code */` to produce the desired return values?

(A) `return x > y;`
(B) `return (x % y) > 1`
(C) `return (x + y) % 2 == 0`
(D) `return (x + y) % x == y;`
(E) `return (x + y) % 2 > (y + y) % 2;`

GO ON TO THE NEXT PAGE

21. Consider the following method that is intended to remove all Strings from `words` that are less than six letters long.

```
            public void letterCountCheck(ArrayList<String> words)
            {
Line 1:        int numWord = 0;
Line 2:        while (numWord <= words.size())
               {
Line 3:           if (words.get(numWord).length() < 6)
                  {
Line 4:              words.remove(numWord);
                  }
                  else
                  {
Line 5:              numWord++;
                  }
               }
            }
```

Which line of code contains an error that prevents `letterCountCheck` from working as intended?

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

22. Consider the following recursive method.

```
public void wackyOutput(String wacky)
{
   if (wacky.length() < 1)
      return;
   wacky = wacky.substring(1, wacky.length());
   wackyOutput(wacky);
   System.out.print(wacky);
}
```

What is printed as a result of executing the call `wackyOutput("APCS")`?

(A) `PC`
(B) `SCPA`
(C) `SCSPCS`
(D) `PCSCSS`
(E) `SCSPCSAPCS`

23. Consider the following method.

```
public String calculate(int num1, int num2)
{
   if (num1 >= 0 && num2 >= 0)
      return "Numbers are valid";
   return "Numbers are not valid";
}
```

Which of the following methods will give the exact same results as `calculate`?

I.
```
public String calculate1(int num1, int num2)
{
   if (!(num1 < 0 || num2 < 0))
      return "Numbers are valid";
   else
      return "Numbers are not valid";
}
```

II.
```
public String calculate2(int num1, int num2)
{
   if (num1 < 0)
      return "Numbers are not valid";
   if (num2 < 0)
      return "Numbers are not valid";
   else
      return "Numbers are valid";
}
```

III.
```
public String calculate3(int num1, int num2)
{
   if (num1 + num2 < 0)
      return "Numbers are not valid";
   return "Numbers are valid";
}
```

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

24. Consider the following class declaration.

```
public class TestClass
{
   private double value;

   public TestClass(double myValue)
   {
      value = myValue;
   }

   public void addValue(double add)
   {
      value += add;
   }

   public void reduceValue(double reduce)
   {
      value -= reduce;
   }

   public double getValue()
   {
      return value;
   }
}
```

The following code segment is executed in the main method.

```
TestClass test1 = new TestClass(9.0);
TestClass test2 = new TestClass(17.5);
test1.reduceValue(3.0);
test2.addValue(1.5);
test2 = test1;
test2.reduceValue(6.0);
System.out.print(test2.getValue() + test1.getValue());
```

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

(A) `0.0`
(B) `3.0`
(C) `12.0`
(D) `19.0`
(E) `27.5`

25. Assume `truth1` and `truth2` are `boolean` variables that have been properly declared and initialized.

Consider this expression.

```
(truth1 && truth2) || ((!truth1) && (!truth2))
```

Which expression below is its logical equivalent?

(A) `truth1 != truth2`
(B) `truth1 || truth2`
(C) `truth1 && truth2`
(D) `!truth1 && !truth2`
(E) `truth1 == truth2`

26. Consider the following method which implements an Insertion Sort algorithm.

```
public void sortIt(int[] arr)
{
   for (int i = 1; i < arr.length; i++)
   {
      int value = arr[i];
      for (int j = i - 1; j >= 0 && reals[j] > value; j--)
      {
         arr[j + 1] = arr[j];
      }
      reals[j + 1] = value;
   }
}
```

Consider the following array which will be passed to the `sortIt` method.

```
int[] arr = {4, 3, 7, 6, 1, 5, 2};
```

What does the array contain after the 3rd time through the outer loop `(i == 3)`?

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

27. Consider the following class.

```
public class Polygon
{
   /** returns true if the ordered pair is inside the polygon
    *
    *          false otherwise
    */
   public boolean contains(int x, int y) // instance variables, constructor, and
                                         other methods not shown
}
```

Consider the following class.

```
public class Rectangle extends Polygon
{
   private int topLeftX, topLeftY, bottomRightX, bottomRightY;

   /**
    *  Precondition: topLeftX < bottomRightX
    *  Precondition: topLeftY < bottomRightY
    *                The y value increases as it moves down the screen.
    *  @param topLeftX The x value of the top left corner
    *  @param topLeftY The y value of the top left corner
    *  @param bottomRightX The x value of the bottom right corner
    *  @param bottomRightY The y value of the bottom right corner
    */
   public Rectangle(int topLX, int topLY, int bottomRX, int bottomRY)
   {
      topLeftX = topLX;
      topLeftY = topLY;
      bottomRightX = bottomRX;
      bottomRightY = bottomRY;
   }
}
```

Which of the following is a correct implementation of the `contains` method?

(A)
```
public boolean contains(int x, int y)
{
   return x > topLeftX && y > topLeftY &&
          x < bottomRightX && y < bottomRightY;
}
```

(B)
```
public boolean contains(int x, int y)
{
   if (topLeftX > x)
      return true;
   else if (topLeftY > y)
      return true;
   else if (bottomRightX < x)
      return true;
   else if (bottomRightY < y)
      return true;
   return false;
}
```

(C)
```
public boolean contains(int x, int y)
{  return super.contains(x, y);  }
```

(D)
```
public boolean contains(int x, int y)
{  return Polygon.contains(x, y);  }
```

(E)
```
public boolean contains(int x, int y)
{
   boolean result = false;
   if (topLeftX < x && topLeftY < y)
      result = true;
   if (bottomRightX > x && bottomRightY > y)
      result = true;
   return result;
}
```

28. Consider the following method.

```
public ArrayList<String> rearrange(ArrayList<String> myList)
{
   for (int i = myList.size() / 2; i >= 0; i--)
   {
      String wordA = myList.remove(i);
      myList.add(wordA);
   }
   return myList;
}
```

Assume that `ArrayList<String>` list has been correctly instantiated and populated with the following entries.

```
["Nora", "Charles", "Madeline", "Nate", "Silja", "Garrett"]
```

What are the values of the `ArrayList` returned by the call `rearrange(list)`?

(A) `["Silja", "Garrett", "Nate", "Madeline", "Charles", "Nora"]`
(B) `["Madeline", "Charles", "Nora", "Nate", "Silja", "Garrett"]`
(C) `["Nate", "Silja", "Garrett", "Nora", "Charles", "Madeline"]`
(D) `["Nora", "Charles", "Madeline", "Nate", "Silja", "Garrett"]`
(E) Nothing is returned. `ArrayListIndexOutOfBoundsException`

29. Consider the following code segment.

```
int varA = -30;
int varB = 30;
while (varA != 0 || varB > 0)
{
   varA = varA + 2;
   varB = Math.abs(varA + 2);
   varA++;
   varB = varB - 5;
}
System.out.println(varA + "    " + varB);
```

What will be printed as a result of executing the code segment?

(A) `-30  30`
(B) `-4  0`
(C) `0  -4`
(D) `0  4`
(E) `-3  -3`

GO ON TO THE NEXT PAGE

30. Consider the following code segment.

```
int[][] grid = { {0,  1,  2},
                 {3,  4,  5},
                 {6,  7,  8},
                 {9, 10, 11} };

int[][] newGrid = new int[grid[0].length][grid.length];

for (int row = 0; row < grid.length; row++)
{
   for (int col = 0; col < grid[row].length; col++)
   {
      newGrid[col][row] = grid[row][col];
   }
}
```

What is the value of `newGrid[2][1]` as a result of executing the code segment?

(A) 3
(B) 4
(C) 5
(D) 7
(E) Nothing is printed. There is an `ArrayIndexOutOfBoundsException`.

31. Consider the following code segment.

```
int[][] grid = new int[3][3];
for (int row = 0; row < grid.length; row++)
{
   for (int col = 0; col < grid[row].length; col++)
   {
      grid[row][col] = 1;
      grid[col][row] = 2;
      grid[row][row] = 3;
   }
}
```

Which of the following shows the values in grid after executing the code segment?

(A)
```
{ {1, 1, 1},
  {1, 1, 1},
  {1, 1, 1} };
```
(B)
```
{ {2, 2, 2},
  {2, 2, 2},
  {2, 2, 2} };
```
(C)
```
{ {3, 2, 2},
  {2, 3, 2},
  {2, 2, 3} };
```
(D)
```
{ {3, 1, 1},
  {1, 3, 1},
  {1, 1, 3} };
```
(E)
```
{ {3, 2, 2},
  {1, 3, 2},
  {1, 1, 3} };
```

32. What can the following method best be described as?

```
public int mystery(int[] array, int a)
{
  for (int i = 0; i < array.length; i++)
  {
    if (array[i] == a)
      return i;
  }
  return -1;
}
```

(A) Insertion Sort
(B) Selection Sort
(C) Binary Search
(D) Merge Sort
(E) Sequential Search

33. The following incomplete code is intended to count the number of times the letter `key` is found in `phrase`.

```
public static int countOccurrences (String phrase, String key)
{
   int count = 0;
   for (int i = 0; i < phrase.length(); i++)
         /*      missing code      */
   return count;
}
```

Which can be used to replace */* missing code */* so that `countOccurrences` works as intended?

(A)
```
if (phrase == key)
        count ++;
```
(B)
```
if (phrase[i].equals(key))
        count ++;
```
(C)
```
if (phrase.indexOf(i).equals(key))
        count ++;
```
(D)
```
if (phrase.substring(i, i+1) == key)
        count ++;
```
(E)
```
if (phrase.substring(i, i+1).equals(key))
        count ++;
```

34. Assume `str1` and `str2` are correctly initialized `String` variables. Which statement correctly prints the values of `str1` and `str2` in alphabetical order?

(A)
```
if (str1 < str2)
    System.out.println(str1 + " " + str2);
else
    System.out.println(str2 + " " + str1);
```
(B)
```
if (str1 > str2)
    System.out.println(str1 + " " + str2);
else
    System.out.println(str2 + " " + str1);
```
(C)
```
if (str1.compareTo(str2))
    System.out.println(str1 + " " + str2);
else
    System.out.println(str2 + " " + str1);
```
(D)
```
if (str1.compareTo(str2) < 0)
    System.out.println(str1 + " " + str2);
else
    System.out.println(str2 + " " + str1);
```
(E)
```
if (str1.compareTo(str2) > 0)
    System.out.println(str1 + " " + str2);
else
    System.out.println(str2 + " " + str1);
```

35. Assume `names` is a `String` array. Which statement correctly chooses a random student's name from the array?

(A) `int student = (int) (Math.random(names.length);`
(B) `int student = (int) (Math.random(names.length - 1));`
(C) `int student = (int) (Math.random() * names.length) + 1;`
(D) `int student = (int) (Math.random() * names.length);`
(E) `int student = (int) (Math.random() * names.length) - 1;`

36. Assume that the variable declarations have been made.

```
int a = 7, b = 15, c = -6;
```

Which will evaluate to true?

(I) `(a < b) || (b < c)`
(II) `!(a < b) && (a < c)`
(III) `(a == b) || ! (b == c)`

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

37. Consider the following method.

```
public int countLetters(String word, String letter)
{
   int count = 0;
   for (int index = 0; index < word.length(); index++)
   {
      if ( /*  missing code  */ )
         count++;
   }
   return count;
}
```

What could replace `/* missing code */` to allow the method to return the number of times `letter` appears in `word`?

(A) `word.substring(index).equals(letter)`
(B) `word.substring(index, index + 1) == letter`
(C) `word.indexOf(letter) == letter.indexOf(letter)`
(D) `word.substring(index, index + 1).equals(letter)`
(E) `letter.equals(word.substring(index).indexOf(letter))`

38. Assume `obscureAnimals` is an `ArrayList<String>` that has been correctly constructed and populated with the following items.

```
["okapi", "aye-aye", "cassowary", "echidna", "sugar glider", "jerboa"]
```

Consider the following code segment.

```
for (int i = 0; i < obscureAnimals.size(); i++)
{
   if (obscureAnimals.get(i).compareTo("pink fairy armadillo") < 0)
      obscureAnimals.remove(i);
}
System.out.print(animals);
```

What will be printed as a result of executing the code segment?

(A) `[]`
(B) `[sugar glider]`
(C) `[aye-aye, echidna, sugar glider]`
(D) `[aye-aye, echidna, sugar glider, jerboa]`
(E) Nothing will be printed. There is an `ArrayListIndexOutOfBoundsException`.

Questions 39–40 refer to the following classes.

Consider the following class declarations.

```
public class Building
{
   private int sqFeet;
   private int numRooms;

   public Building(int ft, int rms)
   {
      sqFeet = ft;
      numRooms = rms;
   }

   public int getSqfeet()
   {  return sqFeet;  }

   public String getSize()
   {
      return numRooms + " rooms and " + sqFeet + " square feet";
   }

   /* Additional implementation not shown */
}

public class House extends Building
{
   private int numBedRooms;
   public House(int ft, int rms, int bedrms)
   {
      super(ft, rms);
      numBedRooms = bedrms;
   }

   public String getSize()
   {
      return super.getSqFeet() + " square feet and " + numBedRooms + " bedrooms.";
   }
}
```

39. Assume that `ArrayList<Building>` list has been correctly instantiated and populated with `Building` objects.

Which of the following code segments will result in the square feet in each building being printed?

I.
```
for (int i = 0; i < list.size(); i++)
    System.out.println(list.getSqFeet(i));
```

II.
```
for (int i = 0; i < list.size(); i++)
    System.out.println(list[i].getSqFeet());
```

III.
```
for (int i = 0; i < list.size(); i++)
    System.out.println(list.get(i).getSqFeet());
```

IV.
```
for (Building b : list)
    System.out.println(list.getSqFeet());
```

V.
```
for (Building b : list)
    System.out.println(b.getSqFeet());
```

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

40. Consider the following code segment.

```
Building b1 = new Building(2000, 3);
Building b2 = new House(2500, 8, 4);
Building b3 = b2;
Building[] buildings = new Building[3];
buildings[0] = b1;
buildings[1] = b2;
buildings[2] = b3;
```

What will be printed by the following code segment?

```
for (int i = 0; i < buildings.length; i++)
   System.out.println(buildings[i].getSize());
```

(A)
```
3 rooms and 2000 square feet
8 rooms and 2500 square feet
8 rooms and 2500 square feet
```

(B)
```
3 rooms and 2000 square feet
2500 square feet and 4 bedrooms
2500 square feet and 4 bedrooms
```

(C)
```
2000 square feet and 3 bedrooms
2500 square feet and 4 bedrooms
2500 square feet and 4 bedrooms
```

(D)
```
3 rooms and 2000 square feet
2500 square feet and 4 bedrooms
3 rooms and 2000 square feet
```

(E) There is an error. Nothing will be printed.

STOP. End of Part I.

GO ON TO THE NEXT PAGE

AP Computer Science A: Practice Exam 2

Part II (Free Response)

Time: 90 minutes
Number of questions: 4
Percent of total score: 50

Directions: Write all of your code in Java. Show all your work.

Notes:

- You may assume all imports have been made for you.
- You may assume that all preconditions are met when making calls to methods.
- You may assume that all parameters within method calls are not null.
- Be aware that you should, when possible, use methods that are defined in the classes provided as opposed to duplicating them by writing your own code.

1. In the seventeenth century, Location Numerals were invented as a new way to represent numbers. Location Numerals have a lot in common with the binary number system we use today, except that in Location Numerals, successive letters of the alphabet are used to represent the powers of two, starting with $A = 2^0$ all the way to $Z = 2^{25}$.

To represent a given number as a Location Numeral (LN), the number is expressed as the sum of powers of two with each power of two replaced by its corresponding letter.

$A = 2^0$ $B = 2^1$ $C = 2^2$ $D = 2^3$ $E = 2^4$ and so on

Let's consider the decimal number 19.

- 19 can be expressed as a sum of powers of 2 like this: $16 + 2 + 1$.
- We can covert 19 to LN notation by choosing the letters that correspond with the powers of 2: EBA.

Here are a few more examples.

Base 10	Sum of Powers	LN
7	$4 + 2 + 1 = 2^2 + 2^1 + 2^0$	CBA
17	$16 + 1 = 2^4 + 2^0$	EA
12	$8 + 4 = 2^3 + 2^2$	DC

A partial `LocationNumeral` class is shown below.

```
public class LocationNumeral
{
   private String letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

   /** Returns the decimal value of a single LN letter
    *
    *  @param letter String containing a single LN letter
    *  @return the decimal value of that letter
    *  Precondition: the parameter contains a single uppercase letter
    */
   public int getLetterValue(String letter)
   {
      /* to be implemented in part (a) */
   }

  /** Returns the decimal value of a Location Numeral
   *
   *  @param numeral String representing a Location Numeral
   *  @return the decimal value of the Location Numeral.
   *  Precondition:  The characters in the parameter are
   *                 uppercase letters A-Z only.
   */
  public int getDecimalValue(String numeral)
  {  /* to be implemented in part (b) */  }
```

```
/** Builds a Location Numeral
*
*  @param value int representing the LN value
*  @return String which represents the LN
*
*  Precondition: 2^0 ≤ value ≤ 2^26
*  Postcondition: The letters in the returned String are in
*                 reverse alphabetical order
*/
public String buildLocationNumeral(int value)

  {  /* to be implemented in part (c) */  }

/* Additional instance variables, constructors, and methods not shown */
}
```

(a) Write the method `getLetterValue` that returns the numerical value of a single LN letter.

The value of each LN letter is equal to 2 raised to the power of the letter's position in the alphabet string.

For example, if passed the letter "E", the method will return 16, which is 2^4.

```
/** Returns the decimal value of a single LN letter
 *
 *  @param letter String containing a single LN letter
 *  @return the decimal value of that letter
 *  Precondition: The parameter contains a single uppercase letter
 */
public int getLetterValue(String letter)
```

(b) Write the method `getDecimalValue` that takes a Letter Numeral and returns its decimal value.

For example, if passed the String "ECA", the method will add the decimal values of E, C, and A and return the result. In this case: 16 + 4 + 1 = 21.

You may assume that `getLetterValue` works as intended, regardless of what you wrote in part (a).

```
/** Returns the decimal value of a simplified Location Numeral
 *
 *  @param numeral String representing a Location Numeral
 *  @return the decimal value of the Location Numeral.
 *  Precondition:  The characters in the parameter are
 *                 uppercase letters A-Z only.
 */
public int getDecimalValue(String numeral)
```

(c) Write the method `buildLocationNumeral` that takes a decimal value and returns the Location Numeral representation of that decimal value.

For example, if passed the value 43, the method will return the `String` "FDBA", which represents the sum 32 + 8 + 2 + 1.

```
/** Builds a Location Numeral
*
*  @param value int representing the LN value
*  @return String which represents the LN
*
*  Precondition:   2^0 ≤ value ≤ 2^26
*  Postcondition: The letters in the returned String are in
*                 descending alphabetical order
*/
public String buildLocationNumeral(int value)
```

2. A quadratic function is a polynomial of degree 2, which can be written in the form $y = ax^2 + bx + c$, where a, b, and c represent real numbers and $a \neq 0$. The x-intercepts, or roots, of a quadratic function can be found by using the quadratic formula.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

The value of the discriminant $\sqrt{(b^2 - 4ac)}$ determines whether the roots are real or non-real (imaginary). If the discriminant > 0, the function has 2 real roots. If the discriminant = 0 the function has 1 real root, and if the discriminant < 0, it has imaginary (0 real) roots. Examples are shown in the table.

Equation	Discriminant	Real Root(s)
$y = 1x^2 - 25$	100.0	5.0, −5.0
$y = 1.2x^2 + 3.6x + 2.7$	0.0	−3.6
$y = 2x^2 - 4.1x + 5.2$	−24.79	none

Assume that the following code segment appears in a class other than `Quadratic`. The code segment shows a sample of using the `Quadratic` class to represent the three equations shown in the table.

```
double discrim, root1, root2;
Quadratic q1 = new Quadratic(1, 0, -25);
discrim = q1.getDiscriminant();      // discrim is assigned 100.0
if (discrim > 0)
{  root1 = q1.root1();               // root1 is assigned 5.0
   root2 = q1.root2();               // root2 is assigned -5.0
}
else if (discrim == 0)
   root1 = q1.root1();

Quadratic q2 = new Quadratic(1.2, 3.6, 2.7);
discrim = q2.getDiscriminant();      // discrim is assigned 0.0
if (discrim > 0)
{  root1 = q2.root1();
   root2 = q2.root2();
}
else if (discrim == 0)
   root1 = q2.root1();               // root1 is assigned -3.6


Quadratic q3 = new Quadratic(2, -4.1, 5.2);
discrim = q3.getDiscriminant();      // discrim is assigned -24.79
if (discrim > 0)
{  root1 = q3.root1();
   root2 = q3.root2();
}
else if (discrim == 0)
   root1 = q3.root1();
```

Write the `Quadratic` class. Your implementation must include a constructor that has three double parameters that represent a, b, and c, in that order. You may assume that the value of a is not zero. It must also include a method `getDiscriminant` that calculates and returns the value of the discriminant, a method `root1` and a method `root2` that will calculate the possible two roots of the equation. Your class must produce the indicated results when invoked by the code segment given above.

3. A printing factory maintains many printing machines that hold rolls of paper.

```
public class Machine
{
   private PaperRoll paper;

   public Machine(PaperRoll roll)
   {  paper = roll;  }

   public PaperRoll getPaperRoll()
   {  return paper;  }

   /** Returns the current partial roll and replaces it with the new roll.
    *  @param pRoll a new full PaperRoll
    *  @return the old nearly empty PaperRoll
    */
   public PaperRoll replacePaper(PaperRoll pRoll)
   {
      /* to be implemented in part (a) */
   }

   /* Additional implementation not shown */
}

public class PaperRoll
{
   private double meters;

   public PaperRoll()
   {  meters = 1000; }

   public double getMeters()
   {  return meters;  }

   /* Additional implementation not shown */
}
```

The factory keeps track of its printing machines in array `machines`, and it keeps track of its paper supply in two lists: `newRolls` to hold fresh rolls and `usedRolls` to hold the remnants taken off the machines when they no longer hold enough paper to be usable. At the beginning of the day, `usedRolls` is emptied of all paper remnants and `newRolls` is refilled. When `newRolls` no longer has enough rolls available to refill all the machines, the factory must shut down for the day until its supplies are restocked. At that time, the amount of paper used for the day is calculated.

A partial `PrintingFactory` class is shown below.

```
public class PrintingFactory
{
   // All machines available in the company
   private Machine[] machines;

   // The available full paper rolls (1000 meters each)
   private ArrayList<PaperRoll> newRolls = new ArrayList<PaperRoll>();

   // The used paper roll remnants (less than 4.0 meters each)
   private ArrayList<PaperRoll> usedRolls = new ArrayList<PaperRoll>();

   public PrintingFactory(int numMachines)
   {   machines = new Machine[numMachines];  }

   /** Replaces the PaperRoll for any machine that has a
    *  PaperRoll with less than 4.0 meters of paper remaining.
    *  The used roll is added to usedRolls.
    *  A new roll is removed from newRolls.
    *  Precondition: newRolls is not empty.
    */
   public void replacePaperRolls(PaperRoll roll)
   {  /* to be implemented in part (b)  */  }

   /** Returns the total amount of paper that has been used.
    *  @return the total amount of paper that has been used from the PaperRolls
    *          in the usedRolls list plus the paper that has
    *          been used from the PaperRolls on the machines.
    */
   public double getPaperUsed()
   {  /* to be implemented in part (c)  */  }

   /* Additional implementation not shown */
}
```

(a) Write the `replacePaper` method of the `Machine` class. This method returns the nearly empty `PaperRoll` and replaces it with the `PaperRoll` passed as a parameter.

```
/** Returns the current partial roll and replaces it with the new roll.
 *  @param pRoll a new full PaperRoll
 *  @return the old nearly empty PaperRoll
 */
public PaperRoll replacePaper(PaperRoll pRoll)
```

(b) Write the `replacePaperRolls` method of the `PrintingFactory` class.

At the end of each job cycle, each `Machine` object in `machines` is examined to see if its `PaperRoll` needs replacing. A `PaperRoll` is replaced when it has less than 4 meters of paper remaining. The used roll is added to the `usedRolls` list, while a new roll is removed from the `newRolls` list.

```
/** Replaces the PaperRoll for any machine in array machines that has
 *  a PaperRoll with less than 4.0 meters of paper remaining.
 *  The used roll is added to usedRolls.
 *  A new roll is removed from newRolls.
 */
public void replacePaperRolls()
```

(c) Write the `getPaperUsed` method of the `PrintingFactory` class.

At the end of the day, the factory calculates how much paper has been used by adding up the amount of paper used from all the rolls in the `usedRolls` list and all the rolls still in the machines. A brand-new paper roll has 1000 meters of paper.

Example

The tables below show the status of the factory at the end of the day.

Paper in Machines	
Amount Left	**Amount Used**
900.0	100.0
250.0	750.0
150.0	850.0

Paper in `usedRolls` List	
Amount Left	**Amount Used**
3.5	996.5
2.0	998.0
1.5	998.5
3.0	997.0

Total used: 100.0 + 750.0 + 850.0 + 996.5 + 998.0 + 998.5 + 997.0 = 5690.0 meters

Complete the method `getPaperUsed`.

```
/** Returns the total amount of paper that has been used.
 *  @return the total amount of paper left on the PaperRolls
 *          in the usedRolls list plus the paper that has
 *          been used from the PaperRolls on the machines.
 */
public double getPaperUsed()
```

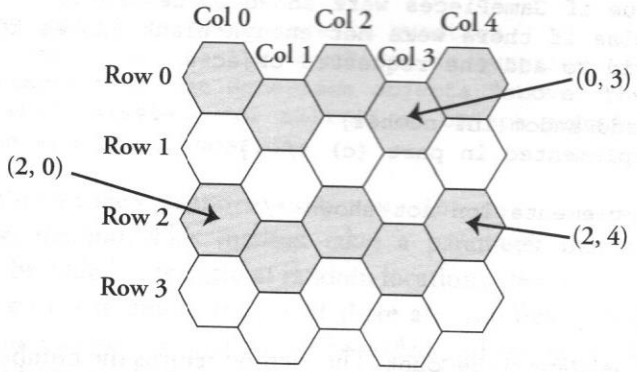
4. A hex grid is a type of two-dimensional (2-D) map used by many games to represent the area that tokens, ships, robots, or other types of game pieces can move around on.

The `GamePiece` class is intended to represent the various game pieces required to play different games on the `HexGrid`. The implementation of the `GamePiece` class is not shown. You may construct a `GamePiece` object with a no-argument constructor.

A hex grid is represented in the `HexGrid` class by a 2-D array of `GamePiece` objects. The coordinates of a `GamePiece` represent its position on the `HexGrid` as shown in the following diagram. Many of the hex cells in the `HexGrid` will not be occupied by any `GamePiece` and are therefore `null`.

The rows in a `HexGrid` are in a staggered horizontal pattern where hexes in even columns rise higher than hexes in odd-numbered columns. The columns in a `HexGrid` always appear vertically aligned regardless of the value of the row.

The following is a representation of a `HexGrid` object that contains several `GamePiece` objects at specified locations in the `HexGrid`. The row and column of each `GamePiece` are given.




```
public class HexGrid
{
  /** A two-dimensional array of the GamePiece objects in the
   *  game. Each GamePiece is located in a specific hex cell
   *  as determined by its row and column number.
   */
  private GamePiece[][] grid;

  /** Returns the number of GamePiece objects
   *  currently occupying any hex cell in the grid.
   *
   *  @return the number of GamePiece objects currently in
   *          the grid.
   */
  public int getGamePieceCount()
  {  /*  to be implemented in part (a) */  }
```

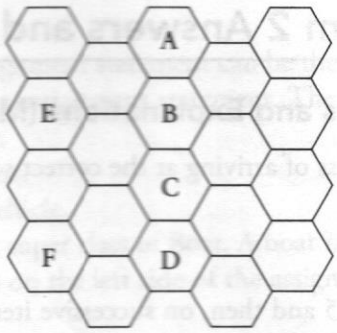
```
    /** Returns an ArrayList of the GamePiece objects
     *  in the same column as and with a lower row number than
     *  the GamePiece at the location specified by the parameters.
     *  If there is no GamePiece at the specified location,
     *  the method returns null.
     *
     *  @param row the row of the GamePiece in question
     *  @param column the column of the GamePiece in question
     *  @return an ArrayList of the GamePiece objects "above" the
     *          indicated GamePiece, or null if no such object
     *          exists
     */
    public ArrayList<GamePiece> isAbove(int row, int col)
    {  /*  to be implemented in part (b) */  }

    /** Adds GamePiece objects randomly to the grid
     *  @ param the number of GamePiece objects to add
     *  @ return true if GamePieces were added successfully
     *          false if there were not enough blank spaces in the
     *          grid to add the requested objects
     */
    public boolean addRandom(int number)
    {  /*  to be implemented in part (c) */  }

    /* Additional implementation not shown */
}
```

(a) Write the method `getGamePieceCount`. The method returns the number of `GamePiece` objects in the hex grid.

```
/** Returns the number of GamePiece objects
 *  currently occupying any hex cell in the grid.
 *
 *  @return the number of GamePiece objects currently in
 *          the grid.
 */
public int getGamePieceCount()
```

(b) Write the method `isAbove`. The method returns an `ArrayList` of `GamePiece` objects that are located "above" the `GamePiece` in the `row` and `column` specified in the parameters. "Above" is defined as in the same column but with a lower row number than the specified location. In the diagram below, letters represent `GamePiece` objects. The objects may appear in any order in the `ArrayList`.

- If the method is called with the row and column of C, an `ArrayList` containing A and B will be returned.
- If the method is called with the row and column of F, an `ArrayList` containing E will be returned.
- If the method is called with the row and column of E, an empty `ArrayList` is returned.
- If the method is called with the row and column of an empty cell, `null` is returned.




```
/** Returns an ArrayList of the GamePiece objects
 *  in the same column as and with a lower row number than
 *  the GamePiece at the location specified by the parameters.
 *  If there is no GamePiece at the specified location,
 *  the method returns null.
 *
 *  @param row the row of the GamePiece in question
 *  @param column the column of the GamePiece in question
 *  @return an ArrayList of the GamePiece objects "above" the
 *          indicated GamePiece, or null if no object exists
 *          at the specified location
 */
public ArrayList<GamePiece> isAbove (int row, int col)
```

(c) Write the `addRandom` method. This method takes a parameter that represents the number of `GamePiece` objects to be added to the grid at random locations. `GamePiece` objects can only be added to locations that are currently empty (`null`). If there are insufficient empty cells to complete the requested additions, no `GamePiece` objects will be added and the method will return `false`. If the additions are successful, the method will return `true`.

You may assume that `getGamePieceCount` works as intended, regardless of what you wrote in part (a). You must use `getGamePieceCount` appropriately in order to receive full credit for part (c).

```
/** Adds GamePiece objects randomly to the grid
 *  @param the number of GamePiece objects to add
 *  @return true if GamePieces were added successfully
 *          false if there were not enough empty cells in the
 *          grid to add the requested objects. In this case
 *          no objects will be added.
 */
public boolean addRandom(int number)
```

STOP. End of Part II.