AP Computer Science A Diagnostic Exam

Part I
Multiple Choice
Time: 45 minutes
Number of questions: 20
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:

- The diagnostic exam is exactly one-half the length of the actual AP Computer Science A Exam.
- You may assume that 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 method.

```
public int someMethod(int val)
{
   for (int i = 2; i < 7; i++)
   {
      if((val + i) % 2 == 0)
      {
         val += 3;
      }
   }
   return val;
}
```

What value is returned by the call `someMethod(13)`?

(A) `17`
(B) `25`
(C) `28`
(D) `31`
(E) Nothing is returned. There is a compile-time error.

2. Consider the following code segment.

```
int num1 = 2;
int num2 = 13;
int result = 4;

if ((num1 < 5) && (num2 < 5))
   result = num1 - num2;
else if ((num1 == 2) && (num2 < 2))
   result = num2 - num1;
else
   result = num1 + num2;
System.out.println(result);
```

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

(A) `-11`
(B) `4`
(C) `11`
(D) `13`
(E) `15`

3. Assume `list` is an `ArrayList<Integer>` that has been correctly constructed and populated with the following items.

```
[13, 7, 0, 5, 12, 6, 10]
```

Consider the following method.

```
public int calculate(ArrayList<Integer> numbers)
{
   int sum = 0;

   for (Integer n : numbers)
   {
      if (n - 8 > 0)
      {
         sum = sum + n;
      }
   }
   return sum;
}
```

What value is returned by the call `calculate(list)`?

(A) `10`
(B) `11`
(C) `13`
(D) `35`
(E) `45`

4. Consider the following class declarations.

```
public class Planet
{
   private String name;
   private double mass;
   private int position;

   public Planet()
   {  /* implementation not shown  */  }

   public Planet(String name)
   {  /* implementation not shown  */  }

   public Planet(String name, int position)
   {  /* implementation not shown  */  }

}

public class DwarfPlanet extends Planet
{
   private double distance;
   public DwarfPlanet(String name)
   {  /* implementation not shown  */  }
}
```

Which of the following declarations compiles without error?

I. `Planet mars = new Planet();`

II. `Planet pluto = new DwarfPlanet("Pluto");`

III. `Planet ceres = new DwarfPlanet();`

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

5. Consider the following code segment.

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

supernatural.add("Vampire");
supernatural.add("Werewolf");
supernatural.add("Ghost");
supernatural.set(0, "Zombie");
supernatural.add(2, "Mummy");
supernatural.add("Witch");
supernatural.remove(3);
System.out.println(supernatural);
```

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

(A) `[Zombie, Werewolf, Mummy, Witch]`
(B) `[Zombie, Werewolf, Ghost, Witch]`
(C) `[Zombie, Werewolf, Mummy, Ghost]`
(D) `[Zombie, Vampire, Werewolf, Mummy, Ghost]`
(E) `[Zombie, Vampire, Werewolf, Mummy, Witch]`

6. Suppose that grid has been initialized as n-by-n 2-D array of integers. Which code segment will add all values that are along the two diagonals?

A.
```
int sum = 0;
for (int row = 0; row < grid.length; row++)
    for (int col = 0; col < grid[0].length; col++)
    {
        sum += grid[row][row];
        sum += grid[row][grid.length-1 - row];
    }
```

B.
```
int sum = 0;
for (int row = 0; row < grid.length; row++)
    for (int col = 0; col < grid[0].length; col++)
    {
        sum += grid[row][row];
        sum += grid[row][grid.length-1 - row];
    }
if(grid.length % 2 == 1)
   sum -= grid[grid.length/2][grid.length/2];
```

C.
```
int sum = 0;
for (int row = 0; row < grid.length; row++)
{
    sum += grid[row][row];
    sum += grid[row][grid.length-1 - row];
}
```

D.
```
int sum = 0;
for (int row = 0; row < grid.length; row++)
{
    sum += grid[row][row];
    sum += grid[row][grid.length-1 - row];
}
sum -= grid[grid.length/2][grid.length/2];
```

E.
```
int sum = 0;
for (int row = 0; row < grid.length; row++)
{
    sum += grid[row][row];
    sum += grid[row][grid.length-1 - row];
}
if(grid.length % 2 == 1)
   sum -= grid[grid.length/2][grid.length/2];
```

7. Consider the following partial class declaration.

```
public class Park
{
   private String name;
   private boolean playground;
   private int acres;

   public Park(String myName, boolean myPlayground, int myAcres)
   {
      name = myName;
      playground = myPlayground;
      acres = myAcres;
   }

   public String getName()
   {  return name;  }

   public boolean hasPlayground()
   {  return playground;  }

   public int getAcres()
   {  return acres;  }

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

Assume that the following declaration has been made in the main method of another class.

```
Park park = new Park("Central", true, 300);
```

Which of the following statements compiles without error?

(A) `int num = park.acres;`

(B) `String name = central.getName();`

(C) `boolean play = park.hasPlayground();`

(D) `int num = park.getAcres(acres);`

(E) `park.hasPlayground = true;`

8. Consider the following code segment.

```
String idk = "mnomnomno";
for (int i = 0; i < idk.length(); i++)
{
   if (idk.substring(i, i + 1).equals("m"))
   {
      idk = idk.substring(0, i) + idk.substring(i + 1, idk.length());
   }
}
System.out.println(idk);
```

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

(A) `mmm`
(B) nonono
(C) `mnomno`
(D) nomnono
(E) `mnomnomno`

9. Consider the following method.

```
public int loopy(int n)
{
   if (n % 7 == 0)
   {
      return n;
   }
   return loopy(n + 3) + 2;
}
```

What value is returned by the call `loopy(12)`?

(A) `12`
(B) `21`
(C) `23`
(D) `27`
(E) `29`

10. Consider the following class declarations.

```
public class Letter
{
   private String letter = "letter";

   public String toString()
   {  return letter; }

   /* Additional implementation not shown */
}

public class ALetter extends Letter
{
   private String letter = "a";

   public String toString()
   {  return letter;  }

   /* Additional implementation not shown */
}

public class BLetter extends Letter
{
   private String letter = "b";

   public String toString()
   {  return letter;  }

   /* Additional implementation not shown */
}

public class CapALetter extends ALetter
{
   private String letter = "A";

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

Consider the following code segment.

```
Letter x = new ALetter();
Letter y = new BLetter();
ALetter z = new CapALetter();

System.out.print(x);
System.out.print(y);
System.out.print(z);
```

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

(A) `abA`

(B) `aba`

(C) `letterlettera`

(D) `letterletterletter`

(E) Nothing is printed. There is a compile-time error.

11. Consider the following method.

```
public String lengthen(String word)
{
   int index = 0;
   while (index < word.length())
   {
      word = word + word.substring(index, index + 1);
      index += 2;
   }
   return word;
}
```

What is returned by the call `lengthen("APCS")`?

(A) `"APCS"`
(B) `"APCSACAA"`
(C) `"APCSAPCS"`
(D) Nothing is returned. Run-time error: `StringIndexOutOfBoundsException`
(E) Nothing is returned. The call will result in an infinite loop.

12. Consider the following code segment.

```
int[] array = {-3, 0, 2, 4, 5, 9, 13, 1, 5};
for (int n = 1; n < array.length - 1; n++)
{
   if (array[n] - array[n - 1] <= array[n] - array[n + 1])
      System.out.print(array[n] + "  ");
}
```

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

(A) `1`
(B) `13`
(C) `13 1`
(D) `2 4 5`
(E) `2 4 5 9`

13. Assume that `k`, `m`, and `n` have been declared and correctly initialized with `int` values. Consider the following statement.

```
boolean b1 = (n >= 4) || ((m == 5 || k < 2) && (n > 12));
```

For which statement below does `b2 = !b1` for all values of `k`, `m`, and `n`?

(A) `boolean b2 = (n >= 4) && ((m == 5 && k < 2) || (n > 12));`
(B) `boolean b2 = (n < 4) || ((m != 5 || k >= 2) && (n <= 12));`
(C) `boolean b2 = (n < 4) && (m != 5) && (k >= 2) || (n <= 12);`
(D) `boolean b2 = (m == 5 || k < 2) && (n > 12);`
(E) `boolean b2 = (n < 4);`

14. Consider the following code segment.

```
int[] ray = new int[11];

for (int i = 0; i < ray.length; i++)
{
   ray[i] = i * 2;
}

for (int m = 0; m < 5; m++)
{
   for (int n = 0; n < 7; n += 2)
   {
      if (m + n > 8)
      {
         System.out.print(ray[m + n]);
      }
   }
}
```

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

(A) Nothing is printed. Runtime error: `ArrayIndexOutOfBounds`
(B) `18`
(C) `181620`
(D) `68`
(E) `1820`

15. Consider the following method.

```
public int mystery(String code, int index)
{
   if (code.indexOf("c") == index)
   {
      return index;
   }
   return mystery(code.substring(2), index + 1);
}
```

Assume that the string `codeword` has been declared and initialized as follows.

```
String codeword = "advanced placement";
```

What value is returned by the call `mystery(codeword, 9)`?

(A) `5`
(B) `6`
(C) `7`
(D) Nothing is returned. Infinite recursion causes a stack overflow error.
(E) Nothing is returned. Run-time error: `StringIndexOutOfBoundsException`

16. Consider the following method.

```
public void switcheroo(int num, int index, int[] nums)
{
    int temp = num;
    num = nums[index];
    nums[index] = temp;
    index++;
}
```

Consider the following code segment.

```
int[] val = {5, 7, 4, -2, 8, 12};
int num = 10;
int index = 3;
switcheroo(num, index, val);

System.out.println("num = " + num + "  val[" + index + "] = " + val[index]);
```

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

(A) `num = 10  val[3] = 10`
(B) `num = 10  val[3] = -2`
(C) `num = 10  val[4] = 8`
(D) `num = -2  val[3] = 10`
(E) `num = -2  val[4] = 8`

17. Consider the following code segment.

```
for (int h = 2; h <= 6; h += 2)
{
   for (int k = 30; k > 0; k -= 10)
   {
      System.out.print(h + k + "    ");
   }
}
```

Consider these additional code segments.

I.
```
int num = 32;
int count = 0;
for (int i = 0; i < 9; i++)
{
   System.out.print(num + "    ");
   num += 2;
   if (count % 3 == 0)
   {
      count = 0;
      num -= 14;
   }
}
```

II.
```
int num = 32;
while (num < 38)
{
   System.out.print(num + "    ");
   num -= 10;
   if (num < 10)
   {
      num += 32;
   }
}
```

III.
```
for (int h = 0; h <= 3; h++)
{
   for (int k = 30; k > 0; k -= 10)
   {
      System.out.print(k + h + "    ");
   }
}
```

Which of the code segments produce the same output as the original code segment?

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

18. Consider the following method.

```
public void mystery (int[] array)
{
   for (int i = 1; i < array.length; i++)
   {
      int j;
      int key = array[i];
      for (j = i - 1; j >= 0 && array[j] > key; j--)
      {
         array[j + 1] = array[j];
      }
      array[j + 1] = key;
   }
}
```

The method above could be best described as an implementation of which of the following?

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

19. Consider the following statement.

```
int number = (int)(Math.random() * 21 + 13);
```

After executing the statement, what are the possible values for the variable `number`?

(A) All integers from 13 to 21 (inclusive).
(B) All real numbers from 13 to 34 (not including 34).
(C) All integers from 13 to 34 (inclusive).
(D) All integers from 13 to 33 (inclusive).
(E) All real numbers from 0 to 21 (not including 21).

20. Consider the following class declaration.

```
public class City
{
   private String name;
   private int population;

   public City(String myName, int myPop)
   {
      name = myName;
      population = myPop;
   }

   public String getName()
   {  return name;  }

   public int getPopulation()
   {  return population;  }

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

Assume `ArrayList<City> cities` has been properly instantiated and populated with `City` objects.

Consider the following code segment.

```
int maxPop = Integer.MIN_VALUE;
for (int i = 0; i < cities.size(); i++)
{
   /* missing code */
}
```

Which of the following should replace `/* missing code */` so that, after execution is complete, `maxPop` will contain the largest population that exists in the `ArrayList`?

(A)
```
City temp = cities[i];
if (temp.getPopulation() > maxPop)
{
   maxPop = temp.getPopulation();
}
```

(B)
```
City temp = cities.get(i);
if (temp.population > maxPop)
{
   maxPop = temp.population;
}
```

(C)
```
if (cities.get(i + 1).getPopulation() > cities.get(i).getPopulation())
{
   maxPop = cities.get(i + 1).getPopulation();
}
```

(D)
```
if (cities.get(i).getPopulation() > maxPop)
{
   maxPop = cities.get(i).getPopulation();
}
```

(E) `maxPop` should have been set to `Integer.MAX_VALUE`. This cannot work as written.

STOP. End of Part I.

AP Computer Science A Diagnostic Exam

Part II
Free Response
Time: 45 minutes
Number of questions: 2
Percent of total score: 50%

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

Notes:

- The diagnostic exam is exactly one-half the length of the actual AP Computer Science A Exam.
- 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. Complex Numbers

In mathematics, a complex number is a number that is composed of both a real component and an imaginary component. Complex numbers can be expressed in the form $a + bi$ where a and b are real numbers and i is the imaginary number $\sqrt{-1}$ (which means that $i^2 = -1$). In complex expressions, a is considered the *real part* and b is considered the *imaginary part.*

Addition with complex numbers involves adding the *real* parts and the *imaginary* parts as two separate sums and expressing the answer as a new complex number.

Assume that the following code segment appears in a class other than `ComplexNumber`. The code segment shows an example of using the `ComplexNumber` class to represent two complex numbers and find their sum.

```
ComplexNumber x1 = new ComplexNumber (3.1, 6.4);
ComplexNumber y1 = new ComplexNumber (-4.8, 2.9);
ComplexNumber z1 = new ComplexNumber ();
z1 = z1.add(x1, y1);
System.out.println(x1 + " + " + y1 + " = " + z1);
     //(3.1 + 6.4i) + (-4.8 + 2.9i) = (-1.7 + 9.3i)

ComplexNumber x2 = new ComplexNumber (3.7, 1);
ComplexNumber y2 = new ComplexNumber (2, -9.2);
ComplexNumber z2 = new ComplexNumber ();
z2 = z2.add(x2, y2);
System.out.println(x2 + " + " + y2 + " = " + z2);
     //(3.7 + 1.0i) + (2.0 + -9.2i) = (5.7 + -8.2i)
```

Write the `ComplexNumber` class. Your implementation must include a constructor that has two `double` parameters that represent a and b, in that order, and a default constructor. It must also include a method `add` that calculates and returns the sum of the two complex numbers represented by its two parameters and a `toString` method that will return a `String` representing the complex number in the form $(a + bi)$. Your class must produce the indicated results when invoked by the code segment given above.

2. Coin Collector

The High School Coin Collection Club needs new software to help organize its coin collections. Each coin in the collection is represented by an object of the `Coin` class. The `Coin` class maintains three pieces of information for each coin: its country of origin, the year it was minted, and the type of coin it is. Because coin denominations vary from country to country, the club has decided to assign a coin type of 1 to the coin of lowest denomination, 2 to the next lowest, and so on. For American coins, `coinType` is assigned like this:

Coin Name	Value	`coinType`
Penny	$0.01	1
Nickel	$0.05	2
Dime	$0.10	3
Quarter	$0.25	4
Half-Dollar	$0.50	5
Dollar	$1.00	6

```
public class Coin
{
   private String country;
   private int year;
   private int coinType;

   public Coin(String cCountry, int cYear, int cType)
   {
      country = cCountry;
      year = cYear;
      coinType = cType;
   }

   public String getCountry()
   {  return country;  }

   public int getYear()
   {  return year;  }

   public int getCoinType()
   {  return coinType;  }
}
```

The Coin Club currently keeps track of its coins by maintaining an `ArrayList` of `Coin` objects for each country. The coins in the `ArrayList` are in order by year, oldest to newest. If two or more coins were minted in the same year, those coins appear in a random order with respect to the other coins from the same year.

The Coin Club has acquired some new collection boxes of various sizes to store their coins. The boxes are rectangular and contain many small compartments in a grid of rows and columns. The club will store coins from different countries in different boxes.

The `CoinCollectionTools` class below assists the Coin Club in organizing and maintaining their collection.

```
public class CoinCollectionTools
{
   private Coin[][] coinBox;

   /** Constructor instantiates coinBox and fills it with
    *  default Coin objects
    *
    *  @param country the country of origin of the coins in this box
    *  @param rows the number of rows in the coin box
    *  @param columns the number of columns in the coin box
    */
   public CoinCollectionTools(String country, int rows, int columns)
   {
      /* to be completed in part (a) */
   }

   /** Creates and returns a completed coinBox grid by adding the
    *  Coin objects from the parameter ArrayList to the coinBox
    *  in column-major order.
    *
    *  @param myCoins the list of Coin objects to be added
    *                 to the coinBox
    *  Precondition:  myCoins is in order by year
    *  Precondition:  the size of myCoins is not greater than the
    *                 the total number of compartments available
    *  @return coinBox the completed 2-D array of Coin objects
    */
   public Coin[][] fillCoinBox(ArrayList<Coin> myCoins)
   {
      /*  to be implemented in part (b)  */
   }

   /** Returns an ArrayList of Coin objects sorted by coinType
    *
    *  @return ArrayList of Coin objects
    *  Precondition: all cells in coinBox contain a valid Coin object
    *  Precondition: coinBox is ordered by year in column-major
    *                order followed by default Coin objects
    *  Postcondition: Coins in ArrayList are in order grouped by coinType
    */
   public ArrayList<Coin> fillCoinTypeList()
   {
      /*  to be implemented in part (c)  */
   }

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

(a) The `CoinCollectionTools` class constructor initializes the instance variable `coinBox` as a two-dimensional array of `Coin` objects with dimensions specified by the parameters. It then instantiates each of the `Coin` objects in the array as a default `Coin` object with `country` equal to the country name passed as a parameter, `year` equal to 0, and `coinType` equal to 0.

Complete the `CoinCollectionTools` class constructor.

```
/** Constructor instantiates coinBox and fills it with
 * default Coin objects
 *
 * @param country the country of origin of the coins in this box
 * @param rows the number of rows in the coin box
 * @param columns the number of columns in the coin box
 */
public CoinCollectionTools(String country, int rows, int columns)
```

(b) The Coin Club intends to fill the collection boxes from their list of coins, starting in the upper-left corner and moving down the columns in order until all `Coin` objects have been placed in a compartment.

The `fillCoinBox` method takes as a parameter an `ArrayList` of `Coin` objects in order by year minted and returns a chart showing their position in the box, filled in column-major order.

You may assume that `coinBox` is initialized as intended, regardless of what you wrote in part (a).

Complete the method `fillCoinBox`.

```
/** Creates and returns a completed coinBox chart by adding the
 *  Coin objects from the parameter ArrayList to the coinBox
 *  in column-major order.
 *
 *  @param myCoins the list of Coin objects to be added
 *                 to the coinBox
 *  Precondition:  myCoins is in order by year
 *  Precondition:  the size of myCoins is not greater than the
 *                 the total number of compartments available
 *  @return coinBox the completed two-dimensional array of Coin objects
 */
public Coin[][] fillCoinBox(ArrayList<Coin> myCoins)
```

(c) Sometimes the Coin Club would prefer to see a list of its coins organized by coin type.

The `fillCoinTypeList` method uses the values in `coinBox` to create and return an `ArrayList` of `Coin` objects filled first with all the `Coin` objects of type 1, then type 2, and so on through type 6. You may assume that no country has more than 6 coin types. Note that the number of coins from any specific type may be 0.

Since the original `coinBox` was filled in column-major order, `Coin` objects should be retrieved from the `coinBox` in column-major order. This will maintain ordering by year within each coin type.

Remember that the `CoinCollectionTools` class constructor filled the `coinBox` with default `Coin` objects with a `coinType` of 0, so no entry in the `coinBox` is `null`.

You may assume that `coinBox` is initialized and filled as intended, regardless of what you wrote in parts (a) and (b).

Complete the method `fillCoinTypeList`.

```
/** Returns an ArrayList of Coin objects sorted by coinType
 *
 *  @return ArrayList of Coin objects
 *  Precondition: all cells in coinBox contain a valid Coin object
 *  Precondition: coinBox is ordered by year in column-major
 *                order followed by default Coin objects
 *  Postcondition: Coins in ArrayList are in order grouped by coinType
 */
public ArrayList<Coin> fillCoinTypeList()
```

STOP. End of Part II.