AP® COMPUTER SCIENCE A
2014 SCORING GUIDELINES

Question 3: Seating Chart

Part (a) `SeatingChart` constructor **5 points**

Intent: *Create* `SeatingChart` *object from list of students*

+1 `seats = new Student[rows][cols];` (or equivalent code)

+1 Accesses all elements of `studentList` *(no bounds errors on* `studentList`*)*

+1 Accesses all necessary elements of seats array *(no bounds errors on seats array, point lost if access not column-major order)*

+1 Assigns value from `studentList` to at least one element in seats array

+1 On exit: All elements of `seats` have correct values (*minor loop bounds errors ok*)

Part (b) `removeAbsentStudents` **4 points**

Intent: *Remove students with more than given number of absences from seating chart and return count of students removed*

+1 Accesses all elements of `seats` (*no bounds errors*)

+1 Calls `getAbsenceCount()` on `Student` object (*point lost if null case not handled correctly)*

+1 Assigns `null` to all elements in seats array when absence count for occupying student > `allowedAbsences` *(point lost if seats array element changed in other cases)*

+1 Computes and returns correct number of students removed

Question-Specific Penalties

-2 (v) Consistently uses incorrect array name instead of `seats` or `studentList`

AP® COMPUTER SCIENCE A
2014 CANONICAL SOLUTIONS

Question 3: SeatingChart

Part (a):

```
public SeatingChart(List<Student> studentList, int rows, int cols){
   seats=new Student[rows][cols];
   int studentIndex=0;
   for (int col = 0; col < cols; col++){
      for (int row = 0; row < rows; row++){
         if (studentIndex < studentList.size()){
            seats[row][col] = studentList.get(studentIndex);
            studentIndex++;
         }
      }
   }
}
```

Part (a) alternate:

```
public SeatingChart(List<Student> studentList, int rows, int cols){
   seats=new Student[rows][cols];
   int row=0;
   int col=0;
   for (Student student : studentList){
      seats[row][col]=student;
      row++;
      if (row==rows){
         row=0;
         col++;
      }
   }
}
```

Part (b):

```
public int removeAbsentStudents(int allowedAbsences){
   int count = 0;
   for (int row=0; row < seats.length; row++){
      for (int col=0; col < seats[0].length; col++){
         if (seats[row][col] != null &&
               seats[row][col].getAbsenceCount() > allowedAbsences){
            seats[row][col]=null;
            count++;
         }
      }
   }
   return count;
}
```

These canonical solutions serve an expository role, depicting general approaches to solution. Each reflects only one instance from the infinite set of valid solutions. The solutions are presented in a coding style chosen to enhance readability and facilitate understanding.

AP® COMPUTER SCIENCE A
2014 SCORING GUIDELINES

Question 4: Trio

Class: `Trio`	**9 points**

Intent: *Define implementation of* `MenuItem` *interface that consists of sandwich, salad, and drink*

+1 `public class Trio implements MenuItem`

+1 Declares appropriate `private` instance variables

+2 Implements constructor

- **+1** `public Trio(Sandwich sandwich, Salad salad, Drink drink)`
- **+1** Initializes appropriate instance variables using parameters

+1 Implements interface methods
(`public String getName(){...}`, `public double getPrice(){...}`)

+1 Constructs correct name string and makes available for return in `getName`

+1 Returns constructed name string in `getName`

+1 Computes correct price and makes available for return in `getPrice`

+1 Returns computed price in `getPrice`

Question-Specific Penalties

-0 Missing or extra spaces in name string, "trio"

-1 (w) Extraneous default constructor that causes side effect

AP® COMPUTER SCIENCE A
2014 CANONICAL SOLUTIONS

Question 4: Trio

```
public class Trio implements MenuItem {
   private Sandwich sandwich;
   private Salad salad;
   private Drink drink;

   public Trio(Sandwich s, Salad sal, Drink d){
      sandwich = s;
      salad = sal;
      drink = d;
   }

   public String getName(){
      return sandwich.getName() + "/" + salad.getName() + "/" +
         drink.getName() + " Trio";
   }

   public double getPrice(){
      double sandwichPrice = sandwich.getPrice();
      double saladPrice = salad.getPrice();
      double drinkPrice = drink.getPrice();
      if (sandwichPrice <= saladPrice && sandwichPrice <= drinkPrice)
         return saladPrice + drinkPrice;
      else if (saladPrice <= sandwichPrice && saladPrice <= drinkPrice)
         return sandwichPrice + drinkPrice;
      else
         return sandwichPrice + saladPrice;
   }
}
```

Alternate

```
public class Trio implements MenuItem {
   private String name;
   private double price;

   public Trio(Sandwich s, Salad sal, Drink d){
      double sandwichPrice = s.getPrice();
      double saladPrice = sal.getPrice();
      double drinkPrice = d.getPrice();
      if (sandwichPrice <= saladPrice && sandwichPrice <= drinkPrice)
         price = saladPrice + drinkPrice;
      else if (saladPrice <= sandwichPrice && saladPrice <= drinkPrice)
         price = sandwichPrice + drinkPrice;
      else
         price = sandwichPrice + saladPrice;
      name = s.getName()+ "/" + sal.getName()+ "/" + d.getName()+ " Trio";
   }
```

These canonical solutions serve an expository role, depicting general approaches to solution. Each reflects only one instance from the infinite set of valid solutions. The solutions are presented in a coding style chosen to enhance readability and facilitate understanding.

Question 4: Trio continued

```
   public String getName(){
      return name;
   }

   public double getPrice(){
      return price;
   }
}
```

These canonical solutions serve an expository role, depicting general approaches to solution. Each reflects only one instance from the infinite set of valid solutions. The solutions are presented in a coding style chosen to enhance readability and facilitate understanding.