2019

AP® CollegeBoard

# AP® Computer Science A

## Sample Student Responses and Scoring Commentary

### Inside:

**Free Response Question 2**

- ☑ **Scoring Guideline**
- ☑ **Student Samples**
- ☑ **Scoring Commentary**

# AP® COMPUTER SCIENCE A
# 2019 SCORING GUIDELINES

Apply the question assessment rubric first, which always takes precedence. Penalty points can only be deducted in a part of the question that has earned credit via the question rubric. No part of a question (a, b, c) may have a negative point total. A given penalty can be assessed only once for a question, even if it occurs multiple times or in multiple parts of that question. A maximum of 3 penalty points may be assessed per question.

**1-Point Penalty**

v) Array/collection access confusion (`[] get`)

w) Extraneous code that causes side-effect (e.g., printing to output, incorrect precondition check)

x) Local variables used but none declared

y) Destruction of persistent data (e.g., changing value referenced by parameter)

z) Void method or constructor that returns a value

**No Penalty**

- Extraneous code with no side-effect (e.g., valid precondition check, no-op)
- Spelling/case discrepancies where there is no ambiguity*
- Local variable not declared provided other variables are declared in some part
- `private` or `public` qualifier on a local variable
- Missing `public` qualifier on class or constructor header
- Keyword used as an identifier
- Common mathematical symbols used for operators (× • ÷ ≤ ≥ <> ≠)
- `[]` vs. `()` vs. `<>`
- `=` instead of `==` and vice versa
- `length` / `size` confusion for array, `String`, `List`, or `ArrayList`; with or without `()`
- Extraneous `[]` when referencing entire array
- `[i,j]` instead of `[i][j]`
- Extraneous size in array declaration, e.g., `int[size] nums = new int[size];`
- Missing `;` where structure clearly conveys intent
- Missing `{ }` where indentation clearly conveys intent
- Missing `( )` on parameter-less method or constructor invocations
- Missing `( )` around `if` or `while` conditions

**Spelling and case discrepancies for identifiers fall under the "No Penalty" category only if the correction can be **unambiguously** inferred from context, for example,* "ArayList" *instead of* "ArrayList". *As a counterexample, note that if the code declares* "`int G=99, g=0;`", *then uses* "`while (G < 10)`" *instead of* "`while (g < 10)`", *the context does **not** allow for the reader to assume the use of the lower-case variable.*

# AP® COMPUTER SCIENCE A
# 2019 SCORING GUIDELINES

## Question 2: Step Tracker

| **Class:** `StepTracker` | **9 points** |
|---|---|

**Intent:** *Define implementation of a class to record fitness data*

**+1** Declares all appropriate `private` instance variables

**+2** Constructor

- **+1** Declares header: `public StepTracker(int ___)`
- **+1** Uses parameter and appropriate values to initialize instance variables

**+3** `addDailySteps` method

- **+1** Declares header: `public void addDailySteps(int ___)`
- **+1** Identifies active days and increments count
- **+1** Updates other instance variables appropriately

**+1** `activeDays` method

- **+1** Declares and implements `public int activeDays()`

**+2** `averageSteps` method

- **+1** Declares header: `public double averageSteps()`
- **+1** Returns calculated `double` average number of steps

# AP® COMPUTER SCIENCE A
# 2019 SCORING GUIDELINES

## Question 2: Scoring Notes

| **Class** `StepTracker` | | | **9 points** |
|---|---|---|---|
| Points | Rubric Criteria | Responses earn the point even if they... | Responses will not earn the point if they... |
| +1 | Declares all appropriate `private` instance variables | | • omit keyword `private`<br>• declare variables outside the class |
| +2 | Constructor | | |
| +1 | Declares header: `public StepTracker(int ___)` | • omit keyword `public` | • declare method `private` |
| +1 | Uses parameter and appropriate values to initialize instance variables | • initialize primitive instance variables to default values when declared | • fail to use the parameter to initialize some instance variable<br>• fail to declare instance variables<br>• initialize local variables instead of instance variables<br>• assign variables to parameters |
| +3 | `addDailySteps` method | | |
| +1 | Declares header: `public void addDailySteps(int ___)` | • omit keyword `public` | • declare method `private` |
| +1 | Identifies active days and increments count | • put valid comparison erroneously in some other method | • fail to use the parameter as part of the comparison<br>• fail to increment a count of active days<br>• fail to increment an instance variable<br>• compare parameter to some numeric constant |
| +1 | Updates other instance variables appropriately | | • update another instance variable only on active days<br>• update another instance variable inappropriately<br>• fail to update appropriate instance variable<br>• update a local variable |
| +1 | `activeDays` method | | |
| +1 | Declares and implements `public int activeDays()` | • return appropriate count of active days where the instance variables were updated improperly in `addDailySteps` or `activeDays` | • declare method `private`<br>• return value that is not the number of active days<br>• fail to return a value |

## Question 2: Scoring Notes (continued)

| Points | Rubric Criteria | Responses earn the point even if they... | Responses will not earn the point if they... |
|---|---|---|---|
| +2 | `averageSteps` method | | |
| +1 | Declares header: `public double averageSteps()` | • omit keyword `public` | • declare method `private` |
| +1 | Returns calculated `double` average number of steps | • maintain instance variables improperly but calculate appropriate average<br>• fail to handle the special case where no days are tracked | • use integer division<br>• calculate something other than steps divided by days<br>• fail to return |

# AP® COMPUTER SCIENCE A
# 2019 SCORING GUIDELINES

## Question 2: Step Tracker

```
public class StepTracker
{
   private int minSteps;
   private int totalSteps;
   private int numDays;
   private int numActiveDays;

   public StepTracker(int threshold)
   {
      minSteps = threshold;
      totalSteps = 0;
      numDays = 0;
      numActiveDays = 0;
   }

   public void addDailySteps(int steps)
   {
      totalSteps += steps;
      numDays++;
      if (steps >= minSteps)
      {
         numActiveDays++;
      }
   }

   public int activeDays()
   {
      return numActiveDays;
   }

   public double averageSteps()
   {
      if (numDays == 0)
      {
         return 0.0;
      }
      else
      {
         return (double) totalSteps / numDays;
      }
   }
}
```

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.

write the complete `StepTracker` class, including the constructor and any required instance variables and methods. Your implementation must meet all specifications and conform to the example.

2A

```
public class StepTracker
{

  int stepGoal;
  double totalSteps;
  int totalDays;
  int totalActiveDays;

  public StepTracker(int stepGoal)
  {
    this.stepGoal = stepGoal; totalSteps = 0; totalDays = 0; totalActiveDays = 0
  }

  public void addDailySteps(int steps)
  {
    totalSteps = totalSteps + steps;
    if (steps >= stepGoal) { totalActiveDays++; }

    totalDays++;
  }

  public int activeDays()
  {
    return totalActiveDays;
  }

  public double averageSteps()
  {
    if (totalDays == 0) { return 0; }
    return (totalSteps / totalDays);
  }

}
```

2B

Write the complete `StepTracker` class, including the constructor and any required instance variables and methods. Your implementation must meet all specifications and conform to the example.

```
public class StepTracker {
    int num = 0; int steps = 0; int days = 0;
    public StepTracker (int activeStepsNum){
      num = activeStepsNum;
    }
    public void addDailySteps (int numOfSteps){
      if (numOfsteps ≥ num){
        steps = steps + numOfSteps;
        days++;
      }
    }
    public int activeDays (){
      return days;
    }
    public double averageSteps (){
      double result = (double)(steps/days);
      return result;
    }
}
```



GO ON TO THE NEXT PAGE.

Write the complete `StepTracker` class, including the constructor and any required instance variables and methods. Your implementation must meet all specifications and conform to the example.

2C

```
public class StepTracker
{ int static ActiveDays;
  int static Ddays;
  int static Dsteps;
  public Tracker (int steps)
  {
   Ddays++;
    Dsteps += steps;
  {
  public void addDailySteps(int Dailysteps)
  {
    if (DailySteps >= 10000)
      {
        ActiveDays++
      }
  }
  public int activeDays()
  {
    return ActiveDays;
  }
public int AverageSteps
{ return Dsteps / Ddays;
}
```

# AP® COMPUTER SCIENCE A
# 2019 SCORING COMMENTARY

## Question 2

**Overview**

This question tested the student's ability to:

- Write program code to define a new type by creating a class; and
- Write program code to satisfy methods using expressions, conditional statements, and iterative statements.

Students were asked to design the class `StepTracker`, which implements a fitness tracking system. Students were expected to demonstrate an understanding of class constructor and method header syntax. Additionally, students were expected to determine the data types and number of instance variables needed to track the information shown in the example. Students were then expected to correctly declare, initialize, access, and update the instance variables. Students were expected to properly protect their data members by declaring them as `private` and to properly define the methods `addDailySteps`, `activeDays`, and `averageSteps`. Students also had to recognize the need for floating-point division when using integers to calculate the average in `averageSteps`.

**Sample: 2A**
**Score: 8**

The response did not earn point 1 because the four instance variables are not declared `private`. The correct constructor header, with an integer parameter, earned point 2. Point 3 was earned by using the parameter and appropriate default values to initialize instance variables. The header for `addDailySteps` is declared appropriately, and point 4 was earned. The parameter of `addDailySteps` is compared to the threshold, and the count is incremented correctly, which earned point 5. Point 6 was earned because the total number of days and steps are updated appropriately. The `activeDays` method has the appropriate header and returns the correct count of active days, so it earned point 7. As the header for `averageSteps` is declared appropriately, point 8 was earned. The response earned point 9 because the average number of steps is correctly calculated and returned.

**Sample: 2B**
**Score: 6**

The response did not declare four `private` instance variables, so it did not earn point 1. The correct constructor header, with an integer parameter, earned point 2. Point 3 was earned by using the parameter to initialize the instance variable. As the header for `addDailySteps` is declared appropriately, point 4 was earned. The parameter of `addDailySteps` is compared to the threshold, and the count is incremented correctly, so point 5 was earned. The response did not earn point 6 because it is missing the increment of the total number of days. In addition, the total number of steps is only incremented in the case of an active day. The `activeDays` method has the appropriate header and returns the correct count of active days, so it earned point 7. As the header for `averageSteps` is declared appropriately, point 8 was earned. The response did not earn point 9 because integer division is performed in the calculation of the average number of steps before casting. In addition, the calculation uses active days instead of total number of days.

## Question 2 (continued)

**Sample: 2C**
**Score: 3**

The response did not earn point 1 because the instance variables are not declared `private`. Point 2 was not earned because the constructor name is incorrect. Point 3 was earned by using the parameter of the constructor to initialize the instance variable. As the header for `addDailySteps` is declared appropriately, point 4 was earned. The response did not earn point 5 because it compares the parameter to a numeric constant. The response omits the update of appropriate instance variables and did not earn point 6. The `activeDays` method has the correct header and returns the appropriate count of active days, so it earned point 7. The response declares the header of `averageSteps` with an `int` return type, so it did not earn point 8. Point 9 was not earned because the response omits casting, causing integer division.