2024

AP® Computer Science A

Sample Student Responses and Scoring Commentary

Inside:

Free-Response Question 1

- ☑ **Scoring Guidelines**
- ☑ **Student Samples**
- ☑ **Scoring Commentary**

Applying the Scoring Criteria

Apply the question scoring criteria 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.*

Question 1: Methods and Control Structures — 9 points

Canonical solution

(a) — **4 points**

```
public void simulateOneDay(int numBirds)
{
   double condition = Math.random();
   if (condition < 0.05)
   {
      currentFood = 0;
   }
   else
   {
      int eachBirdEats = (int) (Math.random() * 41) + 10;
      int totalEaten = numBirds * eachBirdEats;
      if (totalEaten > currentFood)
      {
         currentFood = 0;
      }
      else
      {
         currentFood -= totalEaten;
      }
   }
}
```

(b) — **5 points**

```
public int simulateManyDays(int numBirds, int numDays)
{
   for (int daysSoFar = 0; daysSoFar < numDays; daysSoFar++)
   {
      if (currentFood == 0)
      {
         return daysSoFar;
      }
      simulateOneDay(numBirds);
   }
   return numDays;
}
```

(a) `simulateOneDay`

	Scoring Criteria	Decision Rules	
1	Generates a random value	Responses **can** still earn the point even if they • fail to save or use the generated random value Responses **will not** earn the point if they • fail to call `Math.random`, or possibly the equivalent, at least one time • make any incorrect call to `Math.random`	**1 point**
2	Identifies two cases based on a comparison of a randomly generated value and some constant that implements a 5% probability	Responses **can** still earn the point even if they • reverse the 5/95 probability • compare `double` values using `<=` or `>=` instead of `<` or `>` • incorrectly cast the 5/95 random value, as long as a suitable range is generated and the comparison divides that range appropriately • call `Math.random` incorrectly	**1 point**
3	Generates a random integer that is uniform in the range [10, 50]	Responses **can** still earn the point even if they • call `Math.random` incorrectly	**1 point**
4	Scales for `numBirds` and subtracts appropriate amount from `currentFood` in all cases (*algorithm*)	Responses **can** still earn the point even if they • compare `double` values using `<=` or `>=` instead of `<` or `>` Responses **will not** earn the point if they • reverse the 5/95 probability • incorrectly process the 5/95 range/comparison (e.g., by incorrect casting) • exit the method while `currentFood` has a negative value	**1 point**
		Total for part (a)	**4 points**

(b) `simulateManyDays`

	Scoring Criteria	Decision Rules	
5	Calls `simulateOneDay` with value of `numBirds`	Responses **will not** earn the point if they • fail to make at least one correct call to `simulateOneDay` • make any call to `simulateOneDay` on the class or on an object other than `this` (use of `this` is optional)	**1 point**
6	Loops over the simulation method call and guards that it runs at most `numDays` times	Responses **can** still earn the point even if they • call the method that simulates one day incorrectly • count the simulated days incorrectly, as long as the loop guard would work if the count were corrected • fail to guard against calls to the simulation method when `currentFood <= 0` Responses **will not** earn the point if they • fail to count the simulated days at all	**1 point**
7	Counts the number of times that the method that simulates one day is called with food available in the feeder (*algorithm*)	Responses **can** still earn the point even if they • simulate extra days when `currentFood` is `0`, as long as the count is still correct Responses **will not** earn the point if they • fail to initialize the counter appropriately • compute the correct count but return something else • fail to test if food is available	**1 point**
8	Compares `currentFood` and `0`	Responses **can** still earn the point even if they • fail to make the comparison in the context of a loop • use the result of the comparison incorrectly	**1 point**
9	Returns any `int` value, in all cases	Responses **can** still earn the point even if they • return a constant	**1 point**
		Total for part (b)	**5 points**
		Total for question 1	**9 points**

Alternate canonical for part (b):

```
public int simulateManyDays(int numBirds, int numDays)
{
   int daysSoFar = 0;
   while (currentFood > 0 && daysSoFar < numDays)
   {
      simulateOneDay(numBirds);
      daysSoFar++;
   }

   return daysSoFar;
}
```

Important: Completely fill in the circle that corresponds to the question you are answering on this page.

Question 1 ● Question 2 ○ Question 3 ○ Question 4 ○

Begin your response to each question at the top of a new page.

```
public void simulateOneDay (int numBirds)
{
  double prob = Math.random();
      if(prob <=0.95)
      {
      double food = (Math.random() * 41) + 10;
        int eaten = (int) food * numBirds;
        if( eaten > currentFood)
          {
          currentFood = 0;
          }
          else
          {
          currentFood -= eaten;
          }
        }
      else
      {
       currentFood = 0;
       }

  }
```


Use a pencil only. Do NOT write your name. Do NOT write outside the box.

Important: Completely fill in the circle that corresponds to the question you are answering on this page.

Question 1	Question 2	Question 3	Question 4
●	○	○	○

Begin your response to each question at the top of a new page.

(b)

```
public int simulateManyDays(int numBirds, int numDays)
{
  int days=0;
  int count=0
  for(int
  while (count < numDays)
  {
    simulateOneDay(numBirds);
    if (currentFood >=0)
    {
      days++;
    }
    count++;
  }
  return days+1;

}
```

Important: Completely fill in the circle that corresponds to the question you are answering on this page.

Question 1 ● Question 2 ○ Question 3 ○ Question 4 ○

Begin your response to each question at the top of a new page.

A.)

```
int amt = Math.random() * 10;
currentFood = currentFood - numBirds * amt;
int abnormal = Math.random() * 100;
if (abnormal <= 4){
    currentFood = 0;
}
if (currentFood < 0){
    currentFood = 0;
}
```

B.)

```
for (int x = 0; x <= numDays; x++){
    simulateOneDay(numBirds);
    if(currentFood == 0){
        return x;
    }
}
```

Important: Completely fill in the circle that corresponds to the question you are answering on this page.

Question 1 ● Question 2 ○ Question 3 ○ Question 4 ○

Begin your response to each question at the top of a new page.

A)

```
public void simulateOneDay(int numBirds){
    int random = Math.random(10, 30);
    int num = numBirds * random;
    if (num >= currentFood){
        currentFood = 0;
    }
    else{
        currentFood -= num;
    }
}
```

B)

```
public int simulateManyDays(int numBirds, int numDays){
    int days = 0;
    for(int i = numDays; i >= 0; i--){
        if(currentFood != 0){
            this.simulateOneDay(numBirds);
            days++;
        }
        else{
            return 0;
        }
    return days;
    }
}
```

Question 1

Note: Student samples are quoted verbatim and may contain spelling and grammatical errors.

Overview

This question tested the student's ability to:

- Write program code to create objects of a class and call methods (Skill 3.A).
- Write program code to define a new type by creating a class (Skill 3.B).
- Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements (Skill 3.C).

More specifically, this question assessed the ability to generate random numbers (both to produce probabilities and to generate an integer within a given range), to call instance methods from other instance methods of the same class, to determine when a proper stopping condition has been met, and to report a counted amount.

In part (a) students were asked to determine the amount of food consumed from a bird feeder depending upon a 5% chance of a bear coming along and eating all the food in the feeder. When a bear is present, all the food is consumed, leaving `currentFood` at `0`. In the other case, the birds eat an amount randomly generated in the integer range from 10 to 50 grams. Should there not be enough food for all the birds, the response should guard against consuming more food than is available, leaving `currentFood` at `0` rather than letting it become negative.

In part (b) students were asked to simulate a period of days where the feeder is visited daily, counting and returning the number of days that the food lasts within a given maximum number of days, `numDays`, and with a given regular group of birds, `numBirds`. Students were expected to consider the possibility of starting the test period with no food, running out of food before the end of the test period, or ending the test period with food remaining.

Sample: 1A
Score: 8

In part (a) point 1 was earned because there is a valid call to `Math.random()`. Point 2 was earned because there is a correct implementation of a 5% probability of a bear emptying the feeder and a 95% probability of birds visiting the feeder. The response correctly generates a random number and checks whether the random number is `< 0.95`, in which case birds are feeding at the feeder; otherwise, a bear is at the feeder. The focus of this point is the implementation of the 5% and 95% probabilities. Point 3 was earned because the response generates a random integer on the interval `[10, 50]`. Note that the `double` variable `food` is cast as an `int` when it is used to calculate the total amount eaten by the birds. The focus of point 3 is the generation of the correct integer range, `[10, 50]`. Point 4 was earned because the response subtracts the appropriate amount of food from `currentFood` and guards against the possibility of leaving a negative value in `currentFood`.

In part (b) point 5 was earned because the method `simulateOneDay` is called with `numBirds` as a parameter. Point 6 was earned because the response loops over the simulation `numDays` times.

Question 1 (continued)

Point 7 was not earned because the response returns `1` in the case of an empty feeder when `simulateManyDays` is first invoked, because `simulateManyDays` always returns `days + 1`. Point 8 was earned because `currentFood` is compared with `0`. Point 9 was earned because an `int` value is returned in all cases.

Sample: 1B
Score: 4

In part (a) point 1 was earned because there is a valid call to `Math.random()`. Point 2 was earned because there is a correct implementation of the 5% and 95% probabilities. The response attempts to generate a random number on the interval `[0, 99]` and checks whether the random number is `<= 4`, in which case a bear is at the feeder; otherwise, birds are at the feeder. The failure to cast the generated number as an `int` is assessed in point 4. Point 3 was not earned because the response does not correctly calculate a random integer in the specified range. The code `int amt = Math.random()*40 + 10` attempts to generate a random integer in the desired interval but the point was not earned because the generated random number is not cast as an `int` and the value generated is in the interval `[10, 49]`. Point 4 was not earned because the variable `abnormal` is declared to be an `int` and the randomly generated number is a `double`. Because of the missing cast to `int`, the 5/95 probabilities have not been correctly calculated.

In part (b) point 5 was earned because the method `simulateOneDay` is called with `numBirds` as a parameter. Point 6 was not earned because the upper bound of the `for` loop is `x <= numDays`. This iterates one more than `numDays` times because `x` is initialized to `0`. Point 7 was not earned because the number of simulated days is not counted correctly. In the case where the first call to `simulateOneDay` empties the feeder, the method returns `0` and not the required `1`. Point 8 was earned because `currentFood` is compared with `0`. Point 9 was not earned because the method does not return an `int` value in all cases. There is no provision for returning a value outside of the loop. In the case where `currentFood` is nonzero when the loop ceases execution, no value is returned.

Sample: 1C
Score: 3

In part (a) point 1 was not earned because of an incorrect call to the `Math.random` method. The response incorrectly calls `Math.random` with two parameters. Point 2 was not earned because there is no attempt to identify two cases based upon the 5% probability. Point 3 was not earned because the response does not correctly calculate a random integer in the specified range. The point is also not earned because the generated random number is not cast as an `int`. The incorrect call to the `random` method was assessed in point 1. Point 4 was not earned because the response does not calculate the 5/95 random value.

In part (b) point 5 was earned because `simulateOneDay` is called with `numBirds` as a parameter. The optional use of the `this` reference is valid. Point 6 was not earned because the response incorrectly loops over the simulation `numDays + 1` times. If the condition of the `for` loop had been `i > 0`, the loop would have correctly counted down from `numDays`. Point 7 was not earned because the response does not correctly count the number of days the simulation is called with food available in

Question 1 (continued)

the feeder. The method returns `0` when the feeder is emptied after `simulateOneDay` is called. Point 8 was earned because `currentFood` is compared with `0`. Point 9 was earned because an `int` value is returned in all cases.