Student Performance Q&A:

2006 AP® Computer Science A Free-Response Questions

The following comments on the 2006 free-response questions for AP® Computer Science A were written by the Chief Reader, David Reed of Creighton University in Omaha, Nebraska. They give an overview of each free-response question and of how students performed on the question, including typical student errors. General comments regarding the skills and content that students frequently have the most problems with are included. Some suggestions for improving student performance in these areas are also provided. Teachers are encouraged to attend a College Board workshop to learn strategies for improving student performance in specific areas.

Question 1

What was the intent of this question?

This question focused on abstraction and data structure access. It involved storing and manipulating appointments, each having a time interval associated with it. In part (a) students were required to complete the `conflictsWith` method in the provided `Appointment` class, so that it compared the current appointment with another appointment and determined whether they overlapped. This involved accessing the underlying time interval for the two appointments and calling the appropriate method from the `TimeInterval` class to see if an overlap occurred. A `DailySchedule` class was then provided that stored an `ArrayList` of `Appointment` objects in a private data field. In part (b) students were required to complete the `clearConflicts` method of this class, which involved traversing the `ArrayList`, identifying any appointments that conflicted with the specified appointment (by calling the `conflictsWith` method from part (a)), and removing all conflicting appointments. In part (c) students were required to complete the `addAppt` method, which attempted to add a new appointment to the daily schedule. This involved traversing the `ArrayList` to determine if any conflicts occurred, removing conflicts in the case of an emergency priority, and adding the new appointment as long as no conflicts remained.

How well did students perform on this question?

This question was comparable to A1 questions in previous years in terms of its difficulty. Student performance was reasonable, with the second highest mean score on the exam: 4.34 out of 9 possible points. The distribution of scores was very uniform, with roughly the same number of high (7–9), medium (3–6), and low scores (0–2). There were also very few blanks on this question, suggesting that students at all levels of mastery found some parts of the question that they could complete.

What were common errors or omissions?

Weaker students made the typical `ArrayList` traversal errors—incorrect bounds, off-by-one errors, etc. A common mistake was to treat the `ArrayList` as if it were an array, using `[ ]` for indexing instead of using the `get` method. In addition, confusion with the provided abstractions led to inappropriate method calls of various forms. Some students specified the class name instead of an object when calling a method (e.g., `Appointment.conflictsWith(…) )`, while others called methods on objects of the wrong type (e.g., calling `overlapsWith` on `Appointment` objects). In part (b) a majority of students missed earning all possible points because of incorrect traversals. A simple traversal that starts at the beginning of the list, removes each conflicting appointment, and then increments the traversal index will in fact skip elements after each removal. To avoid this, a correct solution must only increment the index if no removal occurred or, alternatively, traverse the list in reverse order. Many students missed this subtle but essential fact. In part (c) the most common errors involved correctly identifying when the appointment could be added.

Based on your experience of student responses at the AP Reading, what message would you like to send to teachers that might help them to improve the performance of their students on the exam?

Students must be comfortable with interacting classes, including examples such as this one where a simple class is provided and another class stores and manipulates objects of that simple class. This requires keeping track of which methods apply to each type of object and correctly calling those methods on the appropriate object. Students also need to be careful in distinguishing between `ArrayLists` and arrays and recognize that removing an element from an `ArrayList` shifts subsequent elements down and thus may require special care during a traversal.

Question 2

What was the intent of this question?

This question focused on students' ability to design a hierarchy of classes using inheritance. An `Item` interface was provided, along with an abstract `TaxableItem` class that implemented the interface. The `TaxableItem` class contained a private data field for storing a tax rate, a constructor for initializing that field, and an abstract method for accessing its list price. In part (a) students were required to complete the additional `purchasePrice` method, which calculated the purchase price for a `TaxableItem` using its tax rate and list price. In part (b) students were required to design and implement a `Vehicle` class, which was derived from `TaxableItem`. This involved declaring private data fields for storing dealer cost and markup, initializing those fields in a constructor (and using `super` to initialize the tax rate field from `TaxableItem`), implementing the abstract `getListPrice` method, and defining a method for changing the dealer markup.

How well did students perform on this question?

This question was slightly easier than similar inheritance/design questions from previous years. While it did require the students to be comfortable with interfaces, abstract classes, and inheritance, the algorithmic complexity of the solution code was relatively simple. As such, students at all levels of mastery were able to answer parts of this question. Student performance was strong, with the highest mean score on the exam: 4.86 out of 9 possible points. The distribution of scores was fairly uniform, skewed slightly to the high end with very few zeros and blanks.

What were common errors or omissions?

Among weaker students, confusion over the concept of inheritance led to many errors. These students often did not extend the proper class, used the reserved word `implements` rather than `extends`, or incorrectly declared the `Vehicle` class to be `abstract`. Although the course outline speaks to the importance of a class having `private` fields for the aspect of information hiding, the students sometimes declared fields that were `public` or tried to access `private` data fields from the parent class directly instead of calling the appropriate methods. In part (a) many students apparently did not realize that an abstract method could be called by another method and so lost points in failed attempts to get the list price. In part (b) many students lost a point for not recognizing that the constructor must use `super` to initialize the `private` field in the parent class. In addition, points were lost by many students for failing to implement the abstract `getListPrice` method or for overriding the existing `purchasePrice` method.

Based on your experience of student responses at the AP Reading, what message would you like to send to teachers that might help them to improve the performance of their students on the exam?

In general, students need to be more comfortable with inheritance and polymorphism. They need to see examples of inheritance hierarchies and be able to recognize when fields and methods can be inherited and when they need to be overridden. In particular, they need to be aware of the limitations of private fields when implementing inheritance and the use of `super` to call the constructor and methods from a parent class. In addition, the distinctions between an interface and an abstract class need to be clear, and students need to know that abstract methods can be called by nonabstract methods in a class.

Question 3

What was the intent of this question?

This question focused on abstraction, array traversal, and the application of basic algorithms. In part (a) students were given a class to represent customers. The `Customer` class had accessor methods for getting the customer's name and ID, and the students were required to complete the `compareCustomer` method that compared two customers. This involved calling the name and ID accessors on both customers, comparing names (using the `String compareTo` method), and also comparing IDs in the case of identical names. In part (b) students were required to complete a method that took two sorted arrays of `Customers` and merged them into a single array of fixed length. This involved maintaining indexes to the front of the arrays, repeatedly comparing customers from the fronts (using the `compareCustomer` method from part (a)), and copying the "smaller" customer to the merged array.

How well did students perform on this question?

This question was the most algorithmically complex question on the exam. While the array-merging process should have been familiar to students from the context of merge sort, answering this question still required understanding and implementing a complex algorithm that maintained the state of three different arrays. Not surprisingly, overall student performance was weaker than it was on the first two questions, with a mean score of 3.47 out of 9 possible points. The distribution of scores was fairly even, skewed slightly to the low end. Despite the fact that part (a) allowed students to earn points independent of the merge algorithm, the number of zeros and blanks was significantly higher than with the first two questions. This suggests that weaker students may have been scared away or overwhelmed by the question.

What were common errors or omissions?

In part (a) the most common error was in accessing and comparing the name and ID fields of a `Customer`. Solutions needed to call the appropriate accessor methods and then call the appropriate comparators on the results (e.g., call `getName()` on the two customers and then compare the resulting `Strings` using `compareTo`). Some students tried to access the name and ID as if they were public fields or used `<` when comparing `Strings`. In part (b) a significant number of students mistakenly treated the arrays as if they were `ArrayLists`, calling methods such as `get`, `add`, and `remove`. The most costly errors, however, were in the underlying algorithm for merging the arrays. A correct solution required keeping track of the fronts of the two `Customer` arrays and also the number of `Customers` currently in the merged array. Some solution attempts would maintain only one or two counts, making it impossible to maintain the required state. Other failed attempts copied duplicate entries or failed to maintain sorted order when constructing the merged array. It should also be noted that the question explicitly restricted the students from creating auxiliary structures when performing the merge. As a result, students who attempted inefficient solutions, such as appending the two arrays and then sorting, did not receive full credit.

Based on your experience of student responses at the AP Reading, what message would you like to send to teachers that might help them to improve the performance of their students on the exam?

On a timed test, it is difficult to expect students to design and implement complex algorithms. However, questions such as this one, where a complex (but possibly familiar) algorithm is outlined and students are asked to implement it, are likely to appear on future exams. Students should be comfortable reading detailed algorithm descriptions and making the low-level design choices necessary to implement the algorithms. Again, students need to be careful in distinguishing between arrays and `ArrayLists`. Arrays are commonly used when there are a fixed number of items to be stored (as in this question), while `ArrayLists` are commonly used when items are to be added and removed arbitrarily (as in A1).

Question 4

What was the intent of this question?

This question was based on the Marine Biology Simulation (MBS) case study and focused on abstraction and code reuse. Students needed to show their understanding of the case study and its interacting classes in order to implement methods for a particular board game. A `Piece` class was provided for representing game pieces and implemented the `Locatable` interface. The `DropGame` class then represented the board as an `Environment of Pieces`. In part (a) students were required to complete the `dropLocationForColumn` method, which found the `Location` in that column where a dropped piece would rest. This involved traversing the column and identifying the empty location with highest row index. In part (b) students were required to complete the `dropMatchesNeighbors` method, which determined whether a piece dropped in a specified column would result in a win. This involved first identifying the drop location for that `column` (by calling the `dropLocationForColumn` method from part (a)) and then checking neighboring locations for pieces of the same color. This last step could be accomplished in a variety of ways using `Environment`, `Location`, and `Direction` methods.

How well did students perform on this question?

This question had the lowest mean score on the exam: 2.95 out of 9 possible points. This can largely be attributed to an excessive number of zeros and blanks. Typically, the case study question has a large number of blanks, suggesting that some teachers may not be sufficiently emphasizing the case study in exam preparation. The fact that this year's case study question was the last one on the exam increases the number of low scores since some students are rushed or run out of time completely. Ignoring zeros and blanks, however, the distribution of scores was fairly even but still skewed slightly to the low end. This suggests that among the students who knew the case study and had sufficient time performance was reasonable. In fact, if zeros and blanks are disregarded, the mean score for this question was higher than for A3.

What were common errors or omissions?

Common errors on this question included standard coding mistakes as well as errors associated with case study code. In part (a) traversing the column and inspecting each location resulted in many out-of-bounds errors, especially if there were no empty locations in the column. Some students returned the highest empty location instead of the lowest. Other common errors were incorrectly creating a `Location` (e.g., forgetting `new`) and incorrectly accessing the `Environment`. In part (b) errors mainly involved confusion between the `Locations` and the `Pieces` contained in those `Locations`. Many students would access the `Location` and then treat it as if it were the `Piece` (e.g., by calling `color()` on the `Location`). Similarly, when accessing a `Piece` from the `Environment`, many students failed to check for `null` before applying the color method. It is also interesting to note that variables of type `Fish` appeared in many student solutions, even though this particular problem does not involve the `Fish` class at all.

Based on your experience of student responses at the AP Reading, what message would you like to send to teachers that might help them to improve the performance of their students on the exam?

Teachers who are not covering the case study or who are relegating it to the very end of the year need to recognize its importance. There will be a free-response question and several multiple-choice questions based on the case study every year. These questions will depend upon students being familiar with and comfortable using classes from the case study. And, as this question demonstrates, these questions need not use the classes in the same context as the Marine Biology Simulation.