



Student Performance Q&A:

2007 AP[®] Computer Science AB Free-Response Questions

The following comments on the 2007 free-response questions for AP[®] Computer Science AB 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 2-dimensional array access, algorithm implementation, and analysis. Students were provided a class framework for a sliding puzzle board, with a 2-D array field and constructor. In part (a) students were required to complete the `isDone` method, which checks to see if the numbered tiles are in order in the board. This involved traversing the 2-D array, comparing each entry with the previous one (or with a running counter), and ignoring the space. In part (b) students were given an algorithm for initializing a puzzle board and were required to implement that algorithm. This involved repeatedly selecting an `ArrayList` element at random, copying it to the puzzle board, and then removing that element from the `ArrayList`. In parts (c) and (d) students were asked to identify the big-Oh complexity of the described algorithm and a variation in which one step was replaced.

How well did students perform on this question?

This question was slightly easier than 2-D array questions on past exams, due to the pseudocode provided in part (b). Student performance was excellent, with the highest mean score on the exam: 6.37 out of 9. The distribution of scores was skewed to the high end, with many scores in the 6–8 range and very few zeros and blanks. This suggests that stronger students had little trouble with this question and that even weaker students found some parts that they could solve.

What were common student errors or omissions?

The most common errors on this question were in incorrectly identifying the big-Oh efficiency of the algorithms in parts (c) and (d). Answers of $O(n^2)$, $O(n \log n)$, $O(n)$, and $O(\log n)$ were all common, suggesting that some students may have been guessing. In part (a) errors commonly associated with array traversals (e.g., off-by-one errors, invalid indexing) were seen. While the enhanced for loop greatly simplified the traversal, few students chose to use it here. In part (b) the most common errors involved selecting the random index from the `ArrayList`. While a variety of approaches were attempted (using `Math.random`, `Random`, and even `RandNumGenerator` from the case study), errors in calling the methods were frequent.

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?

Some questions on recent exams have focused on “design” issues, where students are required to design and implement a class based on its desired behavior. Another aspect of design is demonstrated here, the ability to analyze the performance of alternative implementations of a solution (either an algorithm or data structure). This aspect may certainly be tested on future exams, so students should be prepared to consider and possibly justify design choices. On a technical note: teachers should realize that the `Random` class has been removed from the APCS Java subset for 2008. Students should be comfortable using `Math.random` to generate random values as needed on future exam questions.

Question 2

What was the intent of this question?

This question focused on abstraction, interacting classes, and data structure use. Students were given two black-box classes: a `Person` class for representing a person and a `Pair` class for representing pairs of persons. In addition, the skeleton of a `PairMatcher` class was provided, which contained a `Map`, mapping a `Person` object to a priority queue of `Pair` objects. In part (a) students were required to implement the constructor for the `PairMatcher` class, which involved initializing the `Map` field, traversing a `List` of `Persons`, adding a `Map` entry for each `Person`, and constructing `Pairs` and inserting them into the `Map`. In part (b) students were required to implement the `removeNumMatches` method, which removes a specified number of `Pairs` from the `Map` (associated with a specific `Person`) and collects them in an array.

How well did students perform on this question?

This question was more difficult than most data-structure oriented questions on previous exams in that it combined a large number of data structures along with interacting classes and a nontrivial algorithm. Student performance was bimodal, with a large number of high scores (7–9) and a similarly high number of zeros and blanks. In between, the scores were fairly evenly distributed. These numbers suggest that strong students were able to handle the complexity of the question and perform well, while the weakest students may have been overwhelmed. The mean for the question was high: 5.34 out of 9. Ignoring the large number of zeros and blanks, the mean rises to a very high 6.49.

What were common student errors or omissions?

While the amount of code required to answer this question was reasonably small, it was highly detailed. In part (a) most students did not instantiate the `personMap` field correctly, either omitting the instantiation completely, attempting to instantiate it as a `Map` (instead of a `HashMap`), or specifying incorrect generic types. Iteration was accomplished in a variety of ways, using iterators, indexing, or enhanced for loops. A common error with the iterator approach was calling the `next` method more than once, and thus skipping entries. In part (b) many students failed to correctly instantiate the array or tried to call the `add` method on the array. Some students were unaware that the elements of a `PriorityQueue` are removed in priority order, and they introduced errors trying to sort the elements.

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

Continue to emphasize the Java Collection classes. Students need to know how to construct collections and use methods to access and update the collections. The Quick Reference Sheet, provided on the exam and available at AP Central, is a valuable reference that students should be encouraged to use (both on the exam and in class). Finally, the difference between interfaces (e.g., `Map` and `List`) and classes (e.g., `HashMap` and `LinkedList`) needs to be better understood by students.

Question 3

What was the intent of this question?

This question focused on creating and recursively traversing a full binary tree of `TreeNode`s. Students were provided the framework of the `GameBoard` class, which had as a field the root of a tree containing integers at the nodes. In part (a) students were required to implement a recursive helper function, which searches all paths in the tree and determines the maximum path sum. This involved recursively finding the maximum path sum for the left and right subtrees, and then adding the root value to that maximum. In part (b) students were required to implement the constructor for the `GameBoard` class, which could be accomplished using either recursion or iteration.

How well did students perform on this question?

This question was comparable in difficulty to previous binary tree questions. Performance was better on part (a), where the recursion was set up and students simply had to implement the helper function. There was greater variability in part (b), since students had to decide whether to use recursion and how to set it up. The distribution of scores was fairly even, with a few more scores at the high end (7–9) than the low end (1–3). There were few zeros and blanks, suggesting that even weaker students had enough familiarity with binary trees to be able to earn some points. The mean for this question was a respectable 4.49 out of 9 (up to 5.13 if zeros and blanks are ignored).

What were common student errors or omissions?

In part (a) more than a third of all students did not properly guard against a null pointer. Either they did not do it at all, did it too late, or only guarded against a child null pointer. Recursively determining the largest subtree sum was problematic for many students, with the most common mistake involving just checking left and right children values. In part (b) most students used recursion although some did attempt an iterative solution, with varying degrees of success. In both approaches, students had difficulty generating a random number in the range 0–9. And while most students could create a `TreeNode` and assign its fields correctly, fully connecting the tree (either recursively or iteratively) was a frequent problem. There was a fundamental lack of understanding when `root` needed to be instantiated and how it worked as a parameter. A common mistake was to just assume that `root` was already instantiated.

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?

Recursion and linked structure manipulation (either linked lists or trees) will continue to be a part of the AB exam, and students need to be comfortable creating and traversing linked structures. Some questions may require recursion explicitly (as in part (a)), while others may present problems that can be solved either recursively or iteratively (as in part (b)). Students should be able to recognize when recursion is a useful problem-solving approach and be able to set up the recursion, possibly defining a recursive helper method. When using recursion to create or modify a tree, students must understand how to return the created/modified subtrees and connect them to the root.

Question 4

What was the intent of this question?

This question was based on the Marine Biology Simulation (MBS) case study and focused on implementing and using an iterator on environments. Students are given the framework of an `EnvIterator` class, which contains fields for storing a `BoundedEnv` and the next `Location` to be returned by the iterator. In part (a) students were required to implement the `next` method, following a detailed description of the order to be taken by the iterator. This could be accomplished by either determining the next location's row and column mathematically, or by traversing along a diagonal until the edge of the environment is reached. In part (b) students were required to implement a method from a client class, which uses an `EnvIterator` to traverse an environment, identify empty locations, and collect them in a `List`.

How well did students perform on this question?

The code required for this question was comparable to questions in previous years, but it did require reading and understanding a significant problem description. Certainly, students who were experienced with using iterators had an advantage, as they were asked to implement the behavior of an iterator over an environment and also use that iterator. The distribution of scores was very uniform, suggesting that students at all levels found aspects of the question that they could implement. The mean of the question

was the lowest on the exam: 4.40 out of 9. However, if zeros and blanks are ignored, the mean was a respectable 5.46 (higher than AB3's mean).

What were common student errors or omissions?

Mapping the diagonal pattern to the actual sequence of locations in part (a) was difficult for many students. Most attempted to directly calculate the correct row and column, following the pseudocode in the problem description, but a significant number instead attempted to traverse the diagonal step-by-step in order to find the next location. This approach often led to errors such as traversing off the edge of the environment. The look-ahead nature of the iterator, where the field stored the next location to be returned instead of the previous one, was confusing to some students who failed to update the field in `next`. Part (b) was much easier for students, although errors in declaring and/or instantiating generic collections were common. Many students were inconsistent in their use of generics or specified the wrong generic types when instantiating. Another common error was attempting to instantiate an interface, as in `new List<Location>()`.

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?

The large number of zeros and blanks on this question could be attributed to several factors: the fact that this was the last question, the complexity of the narrative, or unfamiliarity with the case study. 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. Starting in 2008, familiarity with the new GridWorld case study will be expected of all students. 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. Similar to the Path Finder question from 2006, this question demonstrates that case study questions need not utilize the same context, but instead can use familiar classes from the case study in other application areas.