



Student Performance Q&A:

2005 AP[®] Computer Science AB Free-Response Questions

The following comments on the 2005 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 was based on the Marine Biology Simulation Case Study and focused on abstraction and inheritance. Students needed to show their understanding of the case study and its interacting classes by writing member functions for a new `Salmon` class. In part (a) students were required to override the `nextLocation` method, which selected the next location based on the salmon's age and distance from its home location (both fields of the `Salmon` class). The implementation of this method required the student to call `Fish` methods to access the environment and location, and `Environment` methods to obtain the empty neighbors of the salmon. In part (b) students were required to override the `act` method to produce the desired behavior. This involved inspecting the age of the salmon and its distance from the home location, breeding or moving depending on its state, and incrementing the salmon's age.

How well did students perform on this question?

The question was comparable to last year's case study question in terms of its difficulty. Student performance was excellent: the question had the highest mean score on the exam (6.39 out of a possible 9 points). Scores were heavily skewed to the high end, with large numbers of 7s, 8s, and 9s. The small numbers of 0s and blank answers suggest that even weaker students found this question easy.

What were common student errors or omissions?

The most common error was forgetting "return" in part (a): many students just wrote "super.nextLocation();". Some students mistakenly believed they needed to remove the location behind the fish from consideration as the next location, resulting in a half point deduction. It is interesting to note that students used a variety of algorithms for selecting an empty neighbor. The question required finding an empty neighbor that was closer to home than the current location. While most students simply returned the first such location found, some generated a random number to select from possible candidates, while others found the empty neighbor with least distance to home. Errors in part (b) tended to be minor, such as using `==`, rather than `.equals()` to test for being at the home location. The next most common error was forgetting to age the fish (or not doing it as the last statement executed).

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?

Their strong performance on this question, combined with the small number of blank answers, suggests that Computer Science AB students were familiar with the case study and were capable of using inheritance to build upon existing code. There will be a free-response question and several multiple-choice questions based on the case study every year. Teachers should continue to emphasize the case study, both as a teaching tool and as required background for the exam.

Question 2

What was the intent of this question?

This question focused on students' ability to design a data structure to meet functionality and performance specifications. A class was provided that stored a `HashMap`, mapping postal codes to sets of cities. In part (a) students were required to identify the expected Big-Oh running time of provided methods. In parts (b) and (c) additional functionality was described, and students were asked to identify appropriate data structures, describe their organization, and justify that these structures met performance constraints. Part (d) required implementing code based on the data structure defined in part (b).

While the 2004 exam also included a "design" question, the 2005 question was significantly different. Instead of designing a class hierarchy, students were asked to select from known data structures to design a solution that met constraints. The question combined code, written descriptions, and performance analysis.

How well did students perform on this question?

Overall, student performance was good, especially given the more open-ended nature of this question. The mean score was 5.05 out of a possible 9 points. There were large numbers of 8s and 9s, and the remaining scores were fairly evenly distributed.

What were common student errors or omissions?

There were very few blanks in part (a), as most students at least guessed at the Big-Oh for the methods. Most responses were correct, although $O(N)$ and $O(\log N)$ were common answers. In part (b) most students had the right idea, declaring either a `Map`, `Map` and `Set`, or `Map` and `ArrayList`. Some students recognized that a single `TreeMap` from cities to postal codes would suffice, but many used a `HashMap` and maintained a separate sorted collection of cities (either a sorted `List` or else a `TreeSet`). The correctness of the data structure depended upon the written explanation, as simply identifying a `Set` or `ArrayList` was not sufficient without describing its contents. Grading the written explanations was often difficult, due to vague or unclear language used by students.

Obviously, parts (c) and (d) were dependent upon the answers provided in part (b). In particular, data structures that allowed for the correct execution of `getCodesForCity` but did not meet the $O(\log N)$ constraint lost half a point. A surprising number of students defined an appropriate data structure in part (b) but failed to use it in part (d). Depending on the data structure chosen, some student solutions failed to display the cities in alphabetical order (e.g., by traversing the `keySet` of a `HashMap`) or failed to meet the $O(N)$ constraint (e.g., sorting a `List` of cities).

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?

Design questions such as this one, which require students to make choices in the creation of classes or data structures, are likely to appear on future exams. Students must be familiar with different data structures and their performance and be able to select among them, given program specifications. They also need to be aware that some questions may require descriptive answers, so they must be able to write clear and focused responses.

Question 3

What was the intent of this question?

The question focused on tree manipulation and recursion. An extension to the `TreeNode` class was provided, adding a parent link to the existing class. In part (a) students were required to traverse a binary tree, checking every node to make sure its parent link was correctly assigned. The question strongly suggested using recursion to traverse the tree and check each node's links. In part (b) the concept of a successor node was introduced and an iterative algorithm for finding the successor of a node was described. Students were required to implement this algorithm, handling three possible cases: the node has no successor, the successor is below the node in the tree, and the successor is above the node in the tree.

How well did students perform on this question?

Typically, questions involving trees and recursion are the hardest for students on the Computer Science AB Exam, and that was certainly true this year. The difficulty of this question was compounded by the addition of parent links in tree nodes and the complex algorithm for finding a successor node. Performance was by far the worst on this exam: the mean score was 3.13 out of a possible 9 points.

Scores were heavily skewed to the low end, with very few 9s and 8s. This suggests that even strong students had difficulty. Weak students were able to earn some partial credit, however, resulting in many scores in the 1–3 range.

What were common student errors or omissions?

Clearly, parent links made the tree a more challenging and unfamiliar data structure. A number of students were unable to adapt to this change. Many showed good knowledge of conventional binary search trees. For example, they coded perfectly the statements for finding the smallest node in a binary search tree in part (b), even though they should have just called the `minNode` method provided for them in the problem. It was obvious that they had learned it in class. However, the same students exhibited difficulty in writing code that required the use of parent links to find a successor above `t`.

In general, students did better on part (a) than on part (b). Despite the wording of the question, which suggested a recursive solution, some students attempted iteration. While a few did so correctly, most iterative solutions went far off task. Of those who did attempt recursion, many messed up the base case or failed to traverse more than one path. Both one- and two-parameter helper methods were used. The one-parameter helper method was the prevalent choice, but solutions with a two-parameter helper method (with a node and its parent passed as parameters) appeared to score higher on average.

Part (b) was best done iteratively, without recursion. Some students apparently thought that every tree method should be implemented recursively, so they significantly complicated their code trying to write a recursive version. Most students failed to handle all three successor cases correctly.

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 poor performance on this question suggests that some teachers may have deemphasized linked structures in their classes. While many Java collection classes have been added to the curriculum, it is important to note that students are still expected to construct and manipulate linked lists (using `ListNode`) and trees (using `TreeNode`). For trees, traversals will commonly utilize recursion, although not always (as part (b) demonstrated). Students should be comfortable in creating and tracing recursive methods.

Question 4

What was the intent of this question?

The question focused on manipulating various collections in order to execute the full expansion of an email alias. Aliases and their associated sets of addresses were stored in a provided `Map`. Part (a) involved a straightforward task, implementing a method to append the contents of a `Set` onto the end of a `Queue`. In part (b) students were required to build upon this method to perform the expansion of an email alias. This required constructing a `Queue` containing the alias and then repeatedly expanding the alias at the front of the queue and appending its associated addresses to the end. The addresses themselves had to be collected in a `Set` and returned by the method.

How well did students perform on this question?

Student performance was very good. The question had the second highest mean score on the exam (5.54 out of a possible 9 points), despite a large number of blank answers because it was the last question. Scores were heavily skewed to the high end, with large numbers of 8s and 9s. Scores between 0 and 7 were fairly evenly distributed.

What were common student errors or omissions?

Most responses in part (a) featured the structure of the canonical solution. The most common errors involved incorrect calls of the methods that accessed the structures (e.g., `add` vs. `enqueue`). A few responses misinterpreted the instructions and put elements in the front of the queue rather than the back.

In part (b) most solutions similarly followed the canonical in their structure, but there were also many recursive responses to the question. Some students just used `expandAlias` in place of `addressBook.get`. A few tried to advance through the queue recursively, expecting `expandAlias` to take care of every item in the queue. There were a substantial number of solutions that used a `Set` rather than a `Queue`; most lost usage points for causing the `addressBook` to be altered. When declaring a `Set` or `Queue`, some responses failed to specify a class, such as `HashSet` or `ListQueue`.

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?

Continue to emphasize the collection classes. Students need to know how to construct collections and use methods to access and update the collections. For some questions they will need to identify the Big-Oh performance of basic operations. The difference between interfaces (e.g., `Set` and `List`) and classes (e.g., `HashSet` and `LinkedList`) needs to be better understood by students.