CollegeBoard
Advanced Placement Program

Student Performance Q&A:

2008 AP® Computer Science A Free-Response Questions

The following comments on the 2008 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, `ArrayList` traversal, and the application of basic algorithms. Students were provided with the frameworks of two helper classes: a `Time` class for representing a specific time, and a `Flight` class for representing an airline flight between cities. They were then asked to implement two methods of a third `Trip` class, which stores a sequence of `Flight` objects in an `ArrayList` instance variable. In part (a) students were required to implement the `getDuration` method for determining the length of a trip. This could be accomplished by calling the appropriate `Flight` and `Time` methods on the first and last flights in the `ArrayList` instance variable. In part (b) students were required to implement the `shortestLayover` method for finding the shortest layover between flights on the trip. This involved traversing the `ArrayList` of flights, determining the layover between successive flights (by calling the appropriate `Flight` and `Time` methods), and identifying the minimum layover duration.

How well did students perform on this question?

The question was comparable in difficulty to similar interacting class questions on previous exams. Many scores of 9 and 8 (more than 30 percent combined) suggest that strong students found the question straightforward. This question had the fewest scores of 0 and blank papers (around 18 percent), which suggests that even the weaker students were prepared for this type of problem. Otherwise, the scores were evenly distributed. Overall, the question had the highest mean score on the exam: 4.74 out of a possible 9 points. Disregarding scores of 0 and blank papers, the mean was 5.79.

What were common student errors or omissions?

Most of the student errors on this question involved incorrect data access or method calls. A significant number of students mistakenly used array notation (`[]`) when attempting to `ArrayList` access elements. Some failed to call the `getDeparture` and `getArrival` methods altogether, confusing the `Flight` objects with their time attributes. Interestingly, a small minority of students attempted to construct new `Time` objects in their solutions, even though no specific constructors were provided (nor were they necessary for the problem). The most common algorithmic errors involved incorrect initialization of a running minimum value, such as initializing the minimum to 0 or -1.

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 should expect to see interacting classes and their implementations on future exams. They should be comfortable using the methods of a class, even if they do not know the underlying implementation details. In fact, the ability to look beyond the details and focus on the behavior of provided methods is an important problem-solving skill tested on this exam. As this question demonstrates, common algorithmic tasks, such as finding a minimum value from a list, may be assessed in a variety of contexts.

Question 2

What was the intent of this question?

This question focused on abstraction, string manipulation, `ArrayList` traversal, and algorithm implementation. Students were provided the framework of a `StringPart` class for identifying a substring of a string (by specifying start index and length). Using this idea of a string part, algorithms for encoding and decoding strings as sequences of parts of a master string were described. In part (a) students were required to implement the described algorithm for decoding a string given its representation as an `ArrayList` of `StringPart` objects. This involved traversing the `ArrayList`, accessing the appropriate substrings in the master string (using the `substring` method), and concatenating the substrings to obtain the original string. In part (b) students had to implement the encoding algorithm, which involved constructing an `ArrayList` of `StringPart` objects that represented the given string. A helper method, `findPart`, was provided for extracting the individual string parts, which had to be added to an `ArrayList` in sequence.

How well did students perform on this question?

This was certainly the most difficult question to read and understand on the exam, as the algorithms for encoding and decoding strings were fairly complex and detailed. Not surprisingly, there were a great many scores of 0 and blank papers (more than 34 percent combined), much more than for any other question. This suggests that many students were intimidated by the apparent complexity of the question or simply gave up trying to decipher its meaning. Scores for those who made serious attempts at solutions, however, were evenly distributed. This indicates that the question tested enough different skills to enable students of all levels of mastery to earn some points. The question had the lowest mean score on the exam: 3.36 out of a possible 9 points. Disregarding scores of 0 and blank papers, the mean was a highly respectable 5.11 (higher than question 4's adjusted mean).

What were common student errors or omissions?

As was the case on question 1, many students mistakenly used array notation (`[]`) when attempting to access `ArrayList` elements. In part (b) the most common error by far was incorrect or missing calls to the provided `findPart` method. Despite the fact that the problem clearly states that `findPart` must be used to extract each successive part of the encoding, many students ignored the method altogether or tried to reimplement their own version. Minor errors included not initializing objects (e.g., the `String` and `ArrayList` return values) and using incorrect syntax when calling string methods.

Both parts of the question allowed for a variety of implementation approaches. In part (a) most students used a counter-driven for loop and `ArrayList` indexing to access the individual string parts. This approach led to more minor syntax errors than those made by the students who used the much simpler enhanced for loop. In part (b) student responses fell largely into two different algorithim approaches. Some students used a running index to keep track of how far their string traversal had reached, stopping when the index reached the end of the string. Other students removed the prefix of the string after it had been encoded, resulting in a loop that terminated when the string became empty. Neither approach seemed easier than the other, and student performance was comparable across the two.

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?

Understanding and implementing nontrivial algorithms is an important skill that will be tested on future exams. As was the case with this question, implementing the algorithm may also involve using provided class frameworks and standard data structures (in this case, strings). When the problem statement provides helper methods and states that they must be used in the implementation (as was the case with `findPart` in part [b]), students should recognize that failing to do so can lead to significant penalities. Students should be encouraged to use the enhanced for loop when applicable, as it tends to simplify code and leads to far fewer syntax errors than iterators or counter-driven loops.

Question 3

What was the intent of this question?

This question was based on the GridWorld case study and focused on abstraction and inheritance. Students showed their understanding of the case study and its interacting classes by extending `Critter` to derive an `OpossumCritter` class with modified behavior. In part (a) students were required to override the `processActors` method so that the surrounding neighbors were accessed and the state of the `OpossumCritter` updated according to the characteristics of those neighbors. In part (b) students had to override the `selectMoveLocation` method so that the resulting move (and ultimate survival of the `OpossumCritter`) depended upon its updated state.

How well did students perform on this question?

Students performed well, especially compared to the case study questions of previous years. Even the weaker students who had not mastered the case study in detail were able to earn some points by writing a counter-driven loop and making appropriate conditional checks. In fact, this problem had the second fewest scores of 0 and blank papers (approximately 25 percent combined) on the exam, which is

uncommon for case study questions. Scores were evenly distributed, with an overall mean score of 4.13 out of a possible 9 points. Disregarding scores of 0 and blank papers, the mean was a very strong 5.5.

Students utilized a variety of approaches in part (b). Many wrote code similar to the canonical, relying on the `numStepsDead` instance variable to determine the state of the `OpossumCritter` at each step. As many as half also utilized the color of the `OpossumCritter` (orange = active, black = playing dead) to determine state.

What were common student errors or omissions?

Students often added either unnecessary or incorrect code to their solutions, which typically indicated a lack of understanding of the `Critter` method `act`. For example, in both parts (a) and (b) many students called the `Actor` method `removeSelfFromGrid`, not realizing that doing so violates the postconditions of `processActors` and `selectMoveLocations`. In part (a) some students reimplemented the code from the `getActors` method in their implementation of `processActors`, indicating a lack of understanding of the input parameter. In part (b) more than half of the students misunderstood the nonvoid return type of `super.selectMoveLocation(locs)`, failing to return the value after calling that method.

In general, students seemed to be misreading the problem, and their carelessness caused them to lose credit. Many failed to realize that a neighbor can be neither a friend nor a foe (although this is clearly stated) and therefore called only one of the two methods `isFriend` or `isFoe`. Similarly, the problem statement clearly said that the `OpossumCritter` should play dead if foes outnumber friends, but students often extended this to the case where there were the same number of friends and foes.

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 need to be better aware of the postconditions of each of the five `Critter` methods called inside `act`. In particular, only `processActors` and `makeMove` can change the state of any of the actors in the grid, and the critter itself can only be removed from the grid in `makeMove`. Remind students to read the problem statements carefully, so that they do not overlook any important details.

Question 4

What was the intent of this question?

This question focused on inheritance, class design, and Boolean logic. Students were provided with the `Checker` interface that contains a single `boolean` method named `accept`. In part (a) they were required to design and implement the `SubstringChecker` class (which implements the `Checker` interface) so that the `accept` method returns `true` if its string parameter contains a specific substring. This involved selecting an appropriate instance variable, defining a constructor that takes a `String` as a parameter, and implementing the `accept` method using appropriate `String` methods. In part (b) students were required to implement a different class that implements `Checker`, the `AndChecker` class. This also involved selecting appropriate instance variables, defining a constructor that takes two `Checkers` as parameters, and implementing the `accept` method so that it

calls the `accept` method on both `Checkers` and returns the AND of the two results. In part (c) they were required to complete the construction of a `Checker` object that computed a particular Boolean function.

How well did students perform on this question?

This question was more straightforward than last year's design question. The `StringChecker` class from part (a) had little algorithmic complexity, requiring only a single instance variable, a single assignment in the constructor, and a single call to the `indexOf` method in `accept`. Interestingly, many students tried a more complex approach, making repeated calls to the `substring` method to check each possible match with the goal string. The `AndChecker` class was slightly more complex, requiring instance variables that stored two arbitrary `Checker` objects, but the overall design of the class was similar to part (a). While performance was generally good on parts (a) and (b), few students got full credit on part (c), resulting in very few scores of 9. Scores were evenly distributed between 8 and 1, but there were a large number of scores of 0 and blank papers (more than 30 percent combined). This can partially be explained by the fact that this was the last question on the exam, so some students may simply have run out of time. The mean score was 3.44 out of a possible 9 points. Disregarding scores of 0 and blank papers, the mean rises to 4.94.

What were common student errors or omissions?

Designing and implementing classes from scratch continues to give many students problems. Minor structural errors were common in parts (a) and (b), such as using "extends" instead of "implements," declaring the `accept` method to be public, declaring instance variables to be private, and confusing instance variables with local variables. The most common error in part (b) was declaring the constructor parameters and instance variables to be the wrong type, such as `String` or `SubstringChecker`. In part (c) errors were evenly divided among those involving syntax (e.g., not creating `SubstringCheckers` as needed) and those involving logic (e.g., not combining `AndChecker` and `NotChecker` objects appropriately).

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?

Designing classes from scratch is a skill that will continue to be tested on future exams. As such, students need to be comfortable with the structure of classes and the mechanics of inheritance. Students should know that in a good design, instance variables should be `private` and accessible methods must be `public` (as must any methods required by an interface). Part (c) demonstrates that Boolean logic can be tested using objects other than `boolean` expressions, and students should be prepared to insinuate and combine objects in logical ways.