AP® Computer Science A
2005 Scoring Commentary

The College Board: Connecting Students to College Success

The College Board is a not-for-profit membership association whose mission is to connect students to college success and opportunity. Founded in 1900, the association is composed of more than 4,700 schools, colleges, universities, and other educational organizations. Each year, the College Board serves over three and a half million students and their parents, 23,000 high schools, and 3,500 colleges through major programs and services in college admissions, guidance, assessment, financial aid, enrollment, and teaching and learning. Among its best-known programs are the SAT®, the PSAT/NMSQT®, and the Advanced Placement Program® (AP®). The College Board is committed to the principles of excellence and equity, and that commitment is embodied in all of its programs, services, activities, and concerns.

Visit the College Board on the Web: www.collegeboard.com.
AP Central is the official online home for the AP Program and Pre-AP: apcentral.collegeboard.com.

Question 1

Overview

This question focused on abstraction and data structure access. It involved storing and manipulating a collection of hotel reservations. Students were given a `Reservation` class and the skeleton of a `Hotel` class for representing the collection. The `Hotel` class had two `private` data fields defined, an array of room reservations, and an `ArrayList` to serve as a waiting list when the rooms were full. In part (a) students needed to iterate over the array, searching for an empty room (i.e., an array entry that was `null`). If an empty room were found, then a reservation had to be created and assigned to that array entry. If not, then the customer had to be added to the `ArrayList`. In part (b) students were required to cancel a room reservation and move a customer from the waiting list if possible. This involved calling a method of the `Reservation` class to determine the correct room number, setting the corresponding entry in the `rooms` array to `null`, and then determining whether the `ArrayList` was empty. If not, then the first entry in the `ArrayList` had to be removed and assigned to the `rooms` array.

Sample: 1A
Score: 9

In part (a) the student correctly loops over the `rooms` array and correctly tests each element to see if it is an empty room (`null`). If an empty room is found, the student correctly creates a new reservation using `guestName` and `k`, the index of the empty room. The student also assigns the new reservation to the `null` location in `rooms` and then returns the reservation without completing the loop. If the loop completes, there are no empty rooms so the student adds the `guestName` to `waitList` and returns `null`.

In part (b) the student uses `res.getRoomNumber()` to determine the room number of the canceled reservation and immediately assigns that element in `rooms` to `null`. The student then checks to see if anyone is on the `waitList`. If the `waitList` has entries, the student removes the first name from `waitList` (which also gets the `first` name from `waitList`) and calls `requestRoom` with that name. This method creates the new reservation, assigns it to a `null` location in `rooms`, and returns the newly created reservation. The student immediately returns this result from the current method. If the `waitList` was empty originally, the student returns `null` (the location in `rooms` had already been set to `null`).

Sample: 1B
Score: 6

In part (a) the student correctly loops over the `rooms` array and correctly tests each element to see if it is an empty room (`null`). If an empty room is found, the student attempts to create a new reservation but obtains the room number by calling `getRoomNumber()` on a `null` reference, resulting in loss of the new reservation correctness half point. The student then correctly returns the reservation without completing the loop. The student lost both half points for handling the `waitList` because `waitList.add` and `return null` are within the loop.

Question 1 (continued)

In part (b) the student attempts to use a loop (rather than calling `res.getRoomNumber()`) over the `rooms array` to determine the room number of the canceled reservation and loses the correctness half point since the `if` test may result in a `null pointer exception.` By then assuming that the loop worked correctly, the student assigns the correct location in `rooms` to `null.` After the loop, the student correctly checks to see if anyone is on the `waitList.` If the `waitList` has entries, the student gets the name of the first person. The student lost the half point for creating a new reservation because the room number (`emptyRoom`) is obtained by a call that would result in a `null pointer exception`. The student does not assign the new reservation to any location in `rooms` and does not remove the first name from `waitList` but does return the reservation. If `waitList` was originally empty, the student correctly returns `null` (the correct assign is done in the initial loop).

Sample: 1C
Score: 3

In part (a) the student correctly loops over the `rooms` array and correctly tests each element to see if it is an empty room (`null`). If an empty room is found, the student attempts to create a new reservation but obtains the room number by calling `getRoomNumber()` on a `null` reference, resulting in the loss of the new reservation correctness half point. The reservation is never returned. The loop ends if the end of the array is reached (`x == rooms.length`) or if `hasRoom` is `true`. The latter case caused the student to lose the half point for adding to `waitList` since the guest had already been assigned a room but is still added. There is no `return null`.

In part (b) the student correctly uses `res.getRoomNumber()`. There is no test for an empty `waitList` so the student lost the test half point. This lack of a test caused loss of the get first entry half point because it is done in both the empty and non-empty case. The student reverses the arguments and lost the create new reservation half point and does not assign the reservation to any room. The remove half point was lost due to lack of a `waitList` test. There is no return of the reservation, and without a test, both empty case half points are lost.

Question 2

Overview

This question tested students' ability to design a hierarchy of classes using inheritance. An `abstract Ticket` class was provided, and students were asked to design classes derived from `Ticket` that provided specialized functionality. In part (a) students were required to design and implement a complete class representing `Advance` sales tickets. This involved declaring a `private` field for storing either the price or the number of days in advance that the ticket was purchased. It also involved writing a constructor to initialize that field and overriding the `abstract getPrice` method of the parent class. In part (b) students were asked to design and implement a `StudentAdvance` ticket class, which was derived from `Advance` and gave a special discount for students. This involved writing a constructor and overriding both the `getPrice` and `toString` methods. Since data fields in `Ticket` and `Advance` were `private`, it was necessary to use `super` in the constructor and both methods in order to include the functionality of the parent methods.

Sample: 2A
Score: 8

In part (a) the student has a completely correct solution.

In part (b) the student does not override `getPrice`. The student does call `super.getPrice` and uses it to assign the correct value to the instance variable `price` found in `StudentAdvance`, but confuses the value that would be returned by a `StudentAdvance` object that invokes the method `getPrice`. The student received credit for the call to `super.getPrice`, but lost the half points for the `getPrice` header and return value.

Sample: 2B
Score: 7

In part (a) the student lost a half point for not making `daysInAdvance private` or `protected`. The student declares `cost` as both an instance variable and a local variable within one branch of an `if` statement in `getPrice` and thus lost the half point for the return value from `getPrice`. The code as written contains a syntax error, but even if the statement `double cost = 30` were enclosed in braces or the declaration were moved before the `if` (but still in the braces around the `if`) the `cost` assigned in that branch would not be the `cost` returned.

Part (b) is correct except that `toString` does not call `super.toString()` and attempts to access an inaccessible variable. The student lost the 1½ points for `toString`. The student declares an unused instance variable and then shadows it with a local variable, but in this case it does not cause problems and was not penalized.

Question 2 (continued)

Sample: 2C
Score: 3

In part (a) the student received the half point for the class header, the half point for the constructor header, and the point for assigning data in the constructor. The instance variables are not `private` or `protected`, losing the half point for a `private` data field. Instead of overriding `getPrice`, the student names the method `ticketPrice`, losing the half point for the `getPrice` header. The `ticketPrice` method computes the price correctly but fails to return it, losing the half point for returning the correct value. The incorrect overriding of `toString` was scored in part (b).

In part (b) the student received only the half point for the class header and the half point for the constructor header. There is no attempt to call `super` within the constructor. The `ticketPrice` method does not override `getPrice`, so the student lost the half point for the `getPrice` header. Instead of calling `super.getPrice`, the student compares `numOfDays` to 10, set `price` to 15 or 20, and fails to return it. Even if the `price` were returned it would not receive the half point for returning the correct value, because the 15 and 20 are hard-wired. The question states that if the pricing scheme for `Advance` tickets changes, the `StudentAdvance` price should continue to be computed correctly with no code modifications. This code fails that test. The `toString` method does not call `super.toString()` and attempts to access an inaccessible variable, so received no credit. The student received no credit for these sections and thus is not penalized for failing to declare `numOfDays` or `price` in this class.

Question 3

Overview

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 `ZigZagFish` class. In part (a) students were required to override the `nextLocation` method, which selected the next location following a zig-zag pattern. The implementation of this method required students to utilize `Fish` methods to obtain the environment, location, and direction of the fish, and `Environment` methods to obtain the correct diagonal location and check to see if it were empty. In part (b) students had to override the `move` method to produce the appropriate movement. This involved calling the `nextLocation` method, testing that location, and either moving or changing direction depending on that location.

Sample: 3A
Score: 9

In part (a) the student declares the local variable `zig` inside the scope of the `if` statement. The half point for `isEmpty` was lost because `zig` is not defined outside the scope of the `if` statement. The rest of this student's solution is correct.

Part (b) is completely correct.

Sample: 3B
Score: 6

In part (a) the student correctly determines the environment, current location, and current direction of the fish. The student correctly identifies both forward diagonals and uses the variable `willZigRight` to correctly choose between them. However, the student never checks to determine if the diagonal location is empty, which lost a half point, and since it is possible to return a nonempty diagonal location, the student also lost the return next location half point. The student lost the return current location half point because the current location is never returned.

In part (b) the student correctly calls `nextLocation()`. The student attempts to check if the fish had moved, but checks the wrong location. The student received full credit for reversing the direction of the fish, because it is in the context of checking for movement. The student correctly updates the fish's location, but incorrectly updates `willZigRight` because `willZigRight` is never set back to `true`.

Sample: 3C
Score: 4

In part (a) the student received the half points for accessing the fish's current direction and location. The student earned the location half point even though the call to the location method is missing parentheses.

In part (b) the student lost the half point for correctly checking for movement. The check itself is fine, but the branches of the `if-else` statement are switched. After deducting this half point, the code for updating the fish's location and changing its direction are considered correct. The student also lost 1 point for failing to update the variable `willZigRight`.

Question 4

Overview

This question focused on abstraction, array traversal, and the application of basic algorithms. In part (a) students were required to calculate the average of numbers in a section of an array, given the starting and ending indices. This involved traversing the correct section of the array, summing values, and then dividing by the size of the section. In part (b) students were required to traverse the array and test consecutive items to determine whether the numbers were arranged in increasing order. In part (c) they used the methods they wrote in the previous parts of the question to compute a conditional average, averaging only the last half of the array if the numbers were increasing. This last part focused heavily on abstraction, as code reimplementations received no credit.

Sample: 4A
Score: 9

In part (a) the variable used for the calculation of the sum is initialized correctly. The loop control variable ranges from first to last, inclusive. The sum is calculated correctly. The quotient is calculated and returned. The quotient, a `double` value, is the average with both the correct numerator and denominator.

In part (b) the loop control variable correctly ranges from `0` to `scores.length-2`, which is consistent with its use as indices for `scores`. Consecutive pairs of scores are correctly compared in order to determine if the pair belongs to the set of improving scores. The Boolean value that is calculated and returned correctly differentiates between a collection of scores that is improving and a collection of scores that is not improving.

In part (c) the method `hasImproved` is called correctly. In both cases, the average of the scores is calculated and returned correctly.

Sample: 4B
Score: 7

In part (a) the variable used for the calculation of the sum is initialized correctly. The loop control variable ranges from first to last, inclusive. The sum is calculated correctly. The quotient is calculated and returned. The student received the attempt half point for the calculation of the average but lost the correct half point because the count of the number of scores used in the denominator is incorrect (off by one).

In part (b) the loop control variable correctly ranges from `0` to `scores.length-2`, which is consistent with its use as indices for `scores`. Consecutive pairs of scores are correctly compared in order to determine if the pair belongs to the set of improving scores. The Boolean value that is calculated and returned correctly differentiates between a collection of scores that is improving and a collection of scores that is not improving. While it appears that the Boolean value toggled, the Boolean expression in the `while` loop ensures that it did not toggle.

In part (c) the method `hasImproved` is called correctly. In the case where the scores have improved, the student has reimplemented the `average` method, which did not receive credit since the directions clearly directed the student to call the methods defined in parts (a) and (b). In the other case, the average of the scores is calculated and returned; however, the range of values used is incorrect (the second parameter is off by one).

Question 4 (continued)

Sample: 4C
Score: 3

In part (a) no points were awarded because no loop is shown in the solution. Note that any calculation of the sum must include the values from the `scores` array. The average must include a sum and a count of values from the `scores` array.

In part (b) the student attempts to compare consecutive values in the `scores` array. There is an attempt at the loop, but the loop control variable processes the value of `scores[length]`, which is incorrect. The student attempts to compare consecutive pairs of scores but does not allow for the scores to be equal and thus lost the correctness half point. No points were awarded for calculating and returning the Boolean value, since the value returned is based on a single comparison (last comparison).

In part (c) the method `hasImproved` is called correctly. In both cases, the average of the scores is calculated and returned; however, the second parameter in both calls to `average` is off by one. Therefore the student lost both return correct average half points.