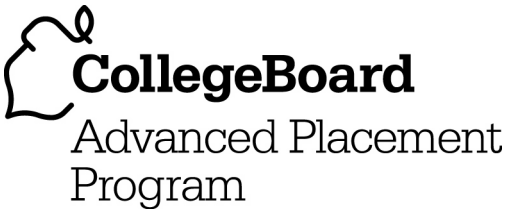

AP® Computer Science A
2004 Scoring Commentary

AP® COMPUTER SCIENCE A
2004 SCORING COMMENTARY

Question 1

Sample: A
Score: 8

This solution lost ½ point in part (a) for failing to return the counter. It also lost the check length correctness ½ point in part (b) for failing to call the length method.

Sample: B
Score: 6

This solution uses array notation throughout which will cause a missed ½ point in parts (a) and (b) on get string. This solution also uses the `.equals` command on parts (a) and (b) which does not work for primitive data types, so it will lose the correctness ½ points in parts (a) and (b) on check length. This solution makes the very common mistake of using a for loop to delete, thereby skipping items.

Sample: C
Score: 3

In both parts, the student used array notation to access an element in `myList` and so loses the correct ½ point for get string. The student also has a single 1-point usage deduction for the `println` statements in both parts. In part (a), the student loses the ½ point initialize counter point because count is not set to 0. The loop correctness is lost in both parts due to the `<=` check. The student also increments `k` twice per loop iteration. In part (a), the student does not attempt to check `len` against anything. There is no `return` in part (a). In part (b), the length check is done correctly (once we assume a string has been gotten). The student tries to remove the item by setting the string to an empty string, but this is not sufficient for even the ½ attempt point on remove.

AP® COMPUTER SCIENCE A
2004 SCORING COMMENTARY

Question 2

Sample: A
Score: 9

Solutions were mostly canonical. All of these declare a local `Pet` variable in part (c), avoiding the need to cast twice.

Sample: B
Score: 6

This was the most commonly-found 6-point solution; its complete absence of `super` kept it from explicitly invoking any of the parent class's methods. It lost points for attempting to access the `Pet` class's private data in both constructors. It lost all `speak` points in part (b). The solution for the `speak` method in part (b) demonstrates a common misunderstanding of the question's illustration ("If the `String` *dog-sound* is returned by the `Dog` method `speak` …"). No deduction was made for missing casts in part (c), which was otherwise correct.

Sample: C
Score: 6

Showing another variation on the constructor problems in parts (a) and (b), this solution declares private instance variables for the pet name and sets them in the constructors, ignoring the inheritance structure. Part (b) has a hard-coded two-word *dog sound* in the `speak` method, losing all associated points. No deduction was made for missing casts in part (c), which was otherwise correct.

Sample: D
Score: 3

Both constructors have incorrect calls to `super`, also attempting to access private `Pet` class data. Part (b)'s `speak` method invokes `Pet`'s `speak` method but neglects to assign the result, losing the correctness ½ point for the returned value. Part (c) earned no points.

AP® COMPUTER SCIENCE A
2004 SCORING COMMENTARY

Question 3

Sample: A
Score: 9

This solution is completely correct. Part (a) implements an alternate algorithm we found. It implements the while loop algorithm with "fish" count incremented one by one until the "fish" density exceeds `minDensity`. Part (b) is a "canonical" solution. Part (c) uses nested for-while loop combination.

Sample: B
Score: 6

Part (a) uses the looping paradigm. However, the loop should use a test for less than or equal to so the answer is sure to exceed the minimum density; this is similar to the rounding error in other papers. The logic in part (b) is correct, but the student does not use `RandNumGenerator.getInstance()` to obtain a random number generator consistent with the case study simulation. Part (c) has a typical wrong answer in which the student uses only a single loop. The student needs to adjust the loop bounds when a fish is not added to the environment. The loop also goes one more time than necessary. Additionally, there is a "double add" where the student calls `theEnv.add()` as well as the `Fish` constructor.

Sample: C
Score: 4

In part (a) the attempt to calculate the total number of fish needed using `minDensity` is off by one. The attempt to return the correct number of additional fish incorrectly uses the `Environment` method `allObjects()`. In part (b) no instance of `Random` is created and the attempts to create a random row and column give values that could be out of bounds. There is no location created. Part (c) has a loop that executes the correct number of times and tries to add the correct number of fish. There is no attempt to generate random locations until one is empty.

Question 4

Sample: A
Score: 9

In part (a), the student correctly identifies both situations in which the robot would be blocked (the two ends of the hall).

In part (b), the robot correctly picks up a single item from the current tile (if there are any). The student then checks again whether the tile is empty, since it might have become empty as a result of the robot picking up an item. If the tile is now empty, the robot moves or turns correctly based on its direction and whether a forward move is blocked.

In part (c), the student correctly calls the relevant methods, `hallIsClear` and `move`, to clear the hall. The student also correctly calculates and returns the number of times the `move` method was called.

Sample: B
Score: 7

In part (a), the student attempts to identify both situations in which the robot would be blocked (the two ends of the hall), but is off by one in checking the high end of the array.

In part (b), the student correctly decrements the number of items from the current tile (if there are any). The student then checks again whether the tile is now empty before attempting to turn or move. The robot moves correctly when not blocked but does not receive either of the two half points for turning, because the lack of assignment means that the direction is not in fact changed.

In part (c), the student receives ½ point for calling the `move` method in the context of a loop and three half points for declaring, initializing, incrementing, and returning a counter that correctly represents the number of times the `move` method was called. The student does not, however, receive the loop points because the code provided reimplements the functionality provided by the `hallIsClear` method defined in the `Robot` class.

Sample: C
Score: 3

In part (a), the student correctly identifies both situations in which the robot would be blocked (the two ends of the hall).

This student does not receive any points for picking up items and moving forward or turning in part (b).

In part (c), the student receives ½ point for calling the `move` method in the context of a loop and three half points for declaring, initializing, incrementing, and returning a counter that correctly represents the number of times the `move` method was called. The student does not, however, receive the loop points because the code does not use the `hallIsClear` method to correctly control the number of times the `move` method is called.