AP® Computer Science A
2000 Scoring Commentary

These materials were produced by Educational Testing Service (ETS), which develops and administers the examinations of the Advanced Placement Program for the College Board. The College Board and Educational Testing Service (ETS) are dedicated to the principle of equal opportunity, and their programs, services, and employment policies are guided by that principle.

The College Board is a national nonprofit membership association dedicated to preparing, inspiring, and connecting students to college and opportunity. Founded in 1900, the association is composed of more than 3,900 schools, colleges, universities, and other educational organizations. Each year, the College Board serves over three million students and their parents, 22,000 high schools, and 3,500 colleges, through major programs and services in college admission, guidance, assessment, financial aid, enrollment, and teaching and learning. Among its best-known programs are the SAT®, the PSAT/NMSQT™, the Advanced Placement Program® (AP®), and Pacesetter®. The College Board is committed to the principles of equity and excellence, and that commitment is embodied in all of its programs, services, activities, and concerns.

AP® COMPUTER SCIENCE A
2000 SCORING COMMENTARY

Question 1

A "standard" one-dimensional array question, this question measured the student's facility with one of the most basic data structures, an array (apvector) of integers (int). In part (a), the student was asked to write a Boolean function to examine the values in the array at a given index and its two neighbors to determine whether that value was a "mode," defined for this problem to be a value in the array greater than either of its neighbors. Most students did quite well on this part.

In part (b), the student was asked to traverse the array to find the mode and return its index, given that the array was uni-modal, with the mode at an interior point. A correct solution traversed the array starting with index 1 and used the function from part (a) to test for the mode. Most students did well, although many made the error of checking the first and/or last indices in the array. Other common errors were to use incorrect logic for the test, such as using an "if … else" in the loop or forgetting to return the resulting index.

In part (c), the student was asked to write a function to print a histogram of the data represented in the array, scaling the bars so that the longest bar, the mode, had a given length. Students found this part somewhat more difficult, although their overall performance was quite good. Students commonly made errors on the formula for scaling the length of a bar using the value of the mode. Some students also tangled the logic of an outer loop traversing the array with an inner loop creating each bar.

Question 2

This question involved reasoning about the AP Computer Science Large Integer Case Study. For part (a), students needed to understand the given helper function `GetDigit` or to understand the lower level representation of digits as characters in order to implement a new Boolean member function `IsOdd`. Most students did very well on this part. Errors usually involved using / instead of % on the test of the digit, or char-to-int conversion errors when students accessed `myDigits` directly instead of using `GetDigit`. Many students correctly checked the least significant digit against the odd digits 1, 3, 5, 7, 9, having been led by the warning that % was not defined for `BigInt` to avoid using it even for int.

In part (b), students used the `BigInt` operators and functions, including `IsOdd` and `DivBy2`, a new member function given for this problem, to implement a specified algorithm for raising a `BigInt` base to a `BigInt` exponent. In general, students did very well on this part, translating the given algorithm into code. The most typical errors were incorrect syntax for the member function calls for `IsOdd` and `DivBy2`. Some students also tangled the logic of the "if" statement inside the loop, either through not fully understanding the given algorithm or by leaving out braces and poor indentation. Almost all students lost a half point by not making copies of the `BigInt` parameters that were passed as const reference and using the copies in the algorithm -- a pattern that was well established in the original `BigInt` code..

Question 3

Question 3 involved working with an abstract data type WordCollection specified by a class declaration. Thus the students had to use the given functions to access the structure. Similar "abstraction" questions have been on past Pascal APCS exams. With C++, the class structure provides a concise and exact way of specifying such an abstraction. A common error that ran through all three parts of this question was for students to attempt to access the WordCollection as if it were an array rather than using the access functions given in the class specification. These students lost credit for not understanding the abstraction, but got some credit for understanding the algorithmic parts of the problem. A second type of error, common to all three parts, was incorrect syntax in the calls to WordCollection member functions. This type of error received a minor penalty as long as the reference to the WordCollection was clear in the call.

In part (a), students were asked to count the number of occurrences in a given WordCollection of a given word. Most students understood the logic of this part quite well. One common error was incorrect bounds on the loop, because the given abstraction specified that the "indices" given to the FindKth function for the class ranged from 1 to Size, inclusive.

Part (b) required the students to write a function that removed duplicates of a given word from a given WordCollection. Common errors were incorrect scanning of the collection, removing all instances of the word, or too few instances. Some students used an incorrect call to the WordCollection member function Remove that had no reference to the instance of the class. A few students erroneously removed all duplicates of all the words in the given WordCollection.

In part (c), students were required to return the most common word in a given WordCollection. This is a standard find-the-maximum algorithm in the context of the abstract data type. Most students did well with the algorithm, sometimes losing points with incorrect use of the abstraction as noted above. Some students also incorrectly updated the variables maintaining the state of the search (the max value and/or the corresponding word or index.)

Question 4

This question, which was the same on both the A and the AB exam, required the student to implement member functions of a class whose specification was given. The problem involved a method of encryption that used a 6-by-6 two-dimensional array of characters (‘A’ through ‘Z’ and ‘0’ through ‘9’ in arbitrary order) as the "key" for the code. In addition, the question used a `struct Point` to encapsulate the row and column of a position in the array. Thus the student needed to understand the indexing for a two-dimensional array (apmatrix), the use of a `struct` to encapsulate a pair of values, and how to build and/or modify apstrings. This was the most difficult question for A students, but it was the easiest overall for the AB students.

In part (a), the student was asked to return the `Point` that encapsulated the coordinates in the array where a given character appeared. This required a search through all the elements of the apmatrix, most commonly done with a pair of nested loops. Most students handled the search well, but often lost points on the construction and return of the `Point struct`.

Question 4 (cont.)

In part (b), the student was asked to write a member function that returned as an apstring the encrypted pair of characters that corresponded to a pair of characters given in an apstring parameter. This involved pulling the characters out of the string, using the function from part (a) to get the coordinates of each character, then, by interchanging coordinates in the specified way, getting the encrypted characters from the array and putting them into an apstring. The most common errors were not in accessing the apmatrix but were in constructing the apstring of two characters by incorrectly assuming some operations defined for apstring would work with characters. Students also often incorrectly indexed the given apstring, typically `pair[1]` and `pair[2]` instead of `pair[0]` and `pair[1]`. Finally, students often had difficulty with the syntax involved with getting the row and column coordinates from the `Point struct` returned by the `GetCoordinates` function written in part (a). As long as there was evidence of the coordinates used as a pair, they were eligible for the algorithm points, after losing points for incorrect `Point`.

In part (c), the students were asked to write a member function that returned as an apstring the encrypted version of a word given as an apstring parameter. They needed to traverse the given string two characters at a time, pulling out each pair of characters, using the function from part (b) to encrypt that pair, then adding that pair to the apstring result. Each pair could be accessed from the given word either by using the `substr` function for apstring, or by accessing the characters and building another two character apstring. Students were provided with a quick reference to all the AP classes and their member functions, so they had a quick lookup for the correct form of a call to `substr`. The most common errors in this part were with the logic of the loop control, failing to index every other character and its successor correctly, and incorrectly constructing the apstring result. Some students also failed to handle the last character in a word with an odd number of characters.

The difficulty in this problem seemed to come from the combination of several structures that students had to deal with together. This integration of concepts is an important aspect of programming that students should be introduced to later in the A course. It also recurs throughout the AB course.