



## AP<sup>®</sup> Computer Science AB 1999 Scoring Guidelines

**The materials included in these files are intended for non-commercial use by AP teachers for course and exam preparation; permission for any other use must be sought from the Advanced Placement Program. Teachers may reproduce them, in whole or in part, in limited quantities, for face-to-face teaching purposes but may not mass distribute the materials, electronically or otherwise. These materials and any copies made of them may not be resold, and the copyright notices must be retained as they appear here. This permission does not apply to any third-party copyrights contained herein.**

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<sup>®</sup>, the PSAT/NMSQT<sup>™</sup>, the Advanced Placement Program<sup>®</sup> (AP<sup>®</sup>), and Pacesetter<sup>®</sup>. 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.

Copyright 2001 by College Entrance Examination Board. All rights reserved.  
Advanced Placement Program and AP are registered trademarks of the College Entrance Examination Board.

AP<sup>®</sup> COMPUTER SCIENCE AB  
1999 SCORING GUIDELINES

Question 1

**Part A:** Quilt::Quilt

- +1/2 read row and column dimensions from inFile  
Note: consistent use of cin, or no attempt to extract from inFile, loses this 1/2 point and both *read from Infile into myBlock[r][c]* 1/2 points
- +1/2 resize matrix appropriately
- +1 loop over rows and columns
  - +1/2 attempt
  - +1/2 correct
- +1 read from inFile into myBlock[r][c] (consistent stream misuse loses both 1/2 points)
  - +1/2 attempt
  - +1/2 correct

**Usage:**

- 1 incorrectly overwrite myRowsOfBlocks/myColsOfBlocks
- 1/2 Bvar >> inFile or var << inFile

**Notes:**

```
>> // skips whitespace
inFile.get( ); // reads and returns next character, including whitespace (can be OK)
inFile.get(ch); // reads next character into ch, including whitespace (can be OK)
inFile.ignore(num, '\n'); // can be OK
getline(inFile, aString); // can be OK
getch( ); // NOT OK
getchar( ); // NOT OK
>> endl; // NOT OK
```

**Part B:** Quilt::PlaceFlipped

- +1/2 attempt from myBlock into qmat
- +1/2 correct (reflection about X or rotation OK, but reflection about Y axis loses this half)

**Part C:** Quilt::QuiltToMat

- +2 declaration/sizing/return of local matrix
  - +1/2 attempt
  - +1 correct
  - +1/2 return matrix (matrix must be declared to get this half)
- +1 loop over entire structure
  - +1/2 attempt
  - +1/2 correct

**AP<sup>®</sup> COMPUTER SCIENCE AB  
1999 SCORING GUIDELINES**

**Question 1 (cont.)**

- +2 PlaceBlock/PlaceFlipped alternation
  - +1 attempt at alternation (within loop context)
  - +1 consistent and correct (matrix must be declared to get this point)

**Notes:**

checkerboard OK  
stripes that result from flipping every other block OK

**AP<sup>®</sup> COMPUTER SCIENCE AB  
1999 SCORING GUIDELINES**

**Question 2**

<b>Part A:</b>	<b>Div2</b> +1/2 init carryDown (must later attempt to assign value based on a digit)  +1 loop over digits +1/2 begin at most significant end (e.g., NumDigits() - 1) +1/2 correct  +1 compute and store current digit (repeatedly)  +1 compute and store carryDown (repeatedly)  +1/2 normalize quotient (must have attempted to change a digit)
<b>Part B:</b>	<b>DivPos (and operator/)</b> In order to receive any points for Part B, must repeatedly - compute the approximate midpoint  OR  - potentially update both bounds based on a comparison  +1/2 declare and init low, high, and mid appropriately  +1/2 loop until quotient found  +1 1/2 compute and store mid +1/2 sum bounds +1/2 divide by 2 and store (integer division is not defined for BigInts)  +1 identify which side of interval to search next  +1 correctly assign to next value of high/low  +1/2 return quotient
<b>Usage</b>	-1/2 improper syntax on Div2 call -1/2 BigInt.NumDigits();

**AP<sup>®</sup> COMPUTER SCIENCE AB  
1999 SCORING GUIDELINES**

**Question 3**

**Part A:** InsertMember

- +1/2 new (can use non-existent default constructor)
- +1/2 attempted assignment to both data fields and/or next
- +1/2 correct assignment of all data members
- +1/2 correct insertion of new node (watch for empty list failure)

**Part B:** CountLevel

- +1/2 temp pointer to traverse
  - +1 list traversal
    - +1/2 attempt
    - +1/2 correct (missing/incorrect init of temp pointer loses this half)
  - +1 count members
    - +1/2 test
    - +1/2 increment count (in context of a test)
- 1/2 return correct accumulated value (missing init of count loses this half)

**Part C:** PrintClubsWithMostInLevel

- +1 loop over all clubs (no OBOB)
- +1 compare and set max as appropriate
  - +1/2 init (watch for unguarded init to 0th element)
  - +1/2 correct (CountLevel must have correct parameters)
- +1 identify appropriate clubs to print
- +1 correctly print only appropriate clubs (only earns 1/2 if missing line break)

**Usage:**

- 1/2 memory leak

**Notes:**

confusion of `->` vs. `.` results in failure to earn associated points rather than usage deduction (*except* in "attempt" and "identify" points)

**AP<sup>®</sup> COMPUTER SCIENCE AB  
1999 SCORING GUIDELINES**

**Question 4**

**Part A:** CountLevel

- +1 null test and return 0 (must have attempt at else)
  - +1/2 attempt
  - +1/2 correct
  
- +2 recursive calls on left/right
  - +1 attempt (must have both calls, "procedures" OK, need 3 parameters of appropriate type)
  - +1 correct (watch for not incrementing level; level++ is incorrect)
  
- +2 compute and return result when name in subtree (note: presence in root is immaterial)
  - +1 attempt (must attempt to verify that name is in subtree)
  - +1 correct
  
- +2 compute and return result when name is NOT in the subtree
  - +1 attempt
    - compare T -> name to someName
    - must attempt to verify that name is NOT in the subtree
  - +1 correct

**Part B:** RootPath

- +1 null test and return 0 (must have attempt at else)
  - +1/2 attempt
  - +1/2 correct
  
- +1 general case
  - +1/2 attempt
    - "procedures" OK
    - need 3 parameters of appropriate type
  - +1/2 correct