AP Computer Science A
2000 Scoring Guidelines

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.

2000 AP® Computer Science

A Question 1

Part A: `IsMode` **2 pts**

+1 attempt (needs at least one on task comparison, `>=` is OK, `||` instead of `&&`)

+1 correct (1 for true and 0 for false is OK)

Note: loop that bears no relation to `k` or that destroys `k` gets no points

Part B: `ModeIndex` **2 pts**

+1 search array
- **+1/2** attempt
- **+1/2** correct (note: any length bound `> data.length()` or no length bound works)

+1 identify and return mode index
- **+1/2** attempt (calls `IsMode` or reimplements it – reimpl must be perfect to get attempt)
- **+1/2** correct

Note: `IsMode` function used as `void` function loses the full identify mode index point
Note: Without a loop, `IsMode` must be called correctly to earn mode index attempt ½ point

Part C: `PrintHistogram` **5 pts**

+1 get value of mode
- **+1/2** attempt (must attempt to find mode index before printing anything)
- **+1/2** correct
  (`k = ModeIndex(data)` loses correct if `data[k]` is not used later in the computation)

+1 scan array
- **+1/2** attempt (must have attempt to scan the data and draw a bar in loop)
- **+1/2** correct

+2 compute correct bar length
- **+1** attempt (must use mode value and `longestBar`)
  (In the absence of a mode value, must use a data element and `longestBar`)
- **+1** correct

+1 draw bar
- **+1/2** attempt (must have a loop)
- **+1/2** correct (must use `barChar` and include `endl`)

NOTE: If `A[k]` is used instead of `data[k]`, it is –1/2 usage, confused id

2000 AP® Computer Science
A Question 2

Part A: `IsOdd` **2 pts**

+1 attempt (must include test of this `BigInt` object's digit(s) and/or value using `/`, `%`, or list of odd or even digits, and intent to return true in some cases, false in others)

+1 correct

Part B: `Power` **7 pts**

+1 declarations and initializations
- **+1/2** product declared as `BigInt` and initialized to `1`
- **+1/2** copies of `exp` and `base` properly declared and initialized

+1 loop with correct bounds

+1 test for odd exponent
- **+1/2** attempt (no pt for use of `%` or `/`, no pt if `IsOdd` used as void function)
- **+1/2** correct

+1 update product correctly (must be inside some test for odd)

+1 update base copy correctly

+1 update exponent copy
- **+1/2** attempt (must have some reference to `DivBy2`)
- **+1/2** correct

+1 return product

Usage (Part B only):

-1/2 call to private member function (e. g. `Normalize()`, `GetDigit()`)

2000 AP® Computer Science
A Question 3

Part A: `Occurrences` **3 pts**

+1 loop over data from 1 to `C.Size()` or until found (Must use lcv in loop)
- **+1/2** attempt (`C.length()` instead of `C.Size()` OK)
- **+1/2** correct

+1 test for match (Must use class notation, not `C[]`)
- **+1/2** attempt
- **+1/2** correct

+1 state: initialize, update (must be inside test), and return count
- **+1/2** attempt (need ongoing updates inside test and at least one of initialize and return)
- **+1/2** correct

Part B: `RemoveDuplicates` **2 pts** No vector notation permitted

+1 attempt (must include loop and attempt to remove)

+1 correct (Must call `Remove` correctly, i.e. `C.Remove(word)`)

Part C: `MostCommon` **4 pts**

+1 check all values (must use lcv in loop)
- **+1/2** attempt (`C.length()` instead of `C.Size()` OK)
- **+1/2** correct (note: must check index `1` to `C.Size()` inclusive)

+1 get count for item (No vector notation permitted)

+1 correct comparison of old maximum with new count and attempt to update old maximum

+1 state: update max , word, and/or index and return word
- **+1/2** attempt (need ongoing updates of value(s) and at least one of initialization or return)
- **+1/2** correct (includes initialization and return; cannot use vector notation)

2000 AP® Computer Science

A Question 4, AB Question 1

Part A: `GetCoordinates` **2 pts**

+1 Find row and column
- **+1/2** attempt (must examine `ch` and `myMat`)
- **+1/2** correct (no loop errors)

+1 construct and return `Point`
- **+1/2** attempt (must try to construct or assign to a `Point`)
- **+1/2** correct (must get attempt to find row and column)

Part B: `EncryptTwo` **4 pts**

+1 get coordinates
- **+1/2** attempt (must: refer to elements of `pair` and use return value as a `Point` throughout rest of part)
- **+1/2** correct

+1 special case(s) – same columns, same rows
- **+1/2** attempt (to check that `int` coordinates lie on a line instead of at opposite corners of a box)
- **+1/2** correct (tests and handles same column case) (note: same rows can be done as general case)

+2 general case
- **+1** attempt (must set elements of an `apstring` using elements of `myMat`)
- **+1** correct (including return)

Part C: `EncryptWord` **3 pts**

+1 loop over pairs
- **+1/2** attempt (must attempt to process pairs of consecutive letters from `word`)
- **+1/2** correct (stay in bounds, appropriate number of iterations)

+2 update result
- **+1/2** form two character `apstring` from consecutive letters from `word` and use later in context of encryption
- **+1/2** call `EncryptTwo()` with parameter, or reimplement perfectly
- **+1/2** correct last char when odd length
- **+1/2** result assembled as an `apstring` in proper order and returned

Usage:

-1 incorrect use of `apstring`

```
char c, d;
apstring s, t(pair);
```

	Correct examples	Incorrect examples
1a.	`s = c; s += d;` *or*	`s = c + d;`
1b.	`s += c; s += d;`	
2.	`t[0] = c; t[1] = d;`	`s[0] = c; s[1] = d;`
3.	`word.substr(k, 2)`	`word.substr(k, k+1)`
4.	`c = word[k];`	`c = word.substr(k, 1);`

-1/2 `Encryptor.myMat` in any part

-1/2 modifying a const parameter (parts B and/or C); deduct at most once

0 confuse `()` and `[]`; confuse `->` and `.`; `apstring<char>`