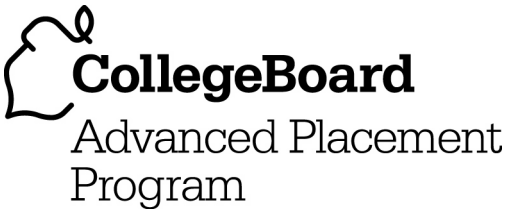

AP® Computer Science A
2004 Scoring Guidelines

AP® Computer Science A
2004 SCORING GUIDELINES

Question 1

Part A: `numWordsOfLength` **4 pts**

+1/2 declare and initialize count to zero (could be an empty list, length = 0)
(must show evidence that variable is used for counting or returned)

+1 loop over `myList`
- **+1/2** attempt (must reference `myList` in body)
- **+1/2** correct

+1/2 get `String` from `myList` (no deduction for missing downcast but local must be `String`)
(lose this if array syntax used)

+1 check length of `String`
- **+1/2** attempt (must be in context of loop)
- **+1/2** correct (array syntax is OK)

+1/2 increment count (must be within context of length check)
(lose this if count does not accumulate)

+1/2 return correct count (after loop is completed)

Part B: `removeWordsOfLength` **5 pts**

+2 loop over `myList`
- **+1** attempt (must reference `myList` in body)
- **+1** correct (must have attempt at removal, must not skip items)

+1 get `String` from `myList` (no deduction for missing downcast, but local must be `String`)
- **+1/2** attempt (must be in context of loop, array syntax is OK)
- **+1/2** correct (no array syntax)

+1 check length of `String`
- **+1/2** attempt (must be in context of loop)
- **+1/2** correct (array syntax is OK)

+1 remove
- **+1/2** attempt (must call `remove`, must refer to `myList` or an index of an element in `myList`)
- **+1/2** correct (no array syntax)

Usage:

-1/2 for `WordList` instead of `myList`

-1/2 for returning a value in part B

-1 for using `this` instead of `myList`, can lose in part A and again in part B (for max of -2)

AP® Computer Science A
2004 SCORING GUIDELINES

Question 2

Part A: `class Cat` **2 pts**

+1/2 `public class Cat extends Pet`

+1/2 Constructor correct (must call `super`)

+1 `speak` method
- **+1/2** attempt (method header matches abstract method, OK if `abstract` left in)
- **+1/2** correct

Part B: `class LoudDog` **3 pts**

+1/2 `public class LoudDog extends Dog`

+1 Constructor correct (must call `super`)

+1 1/2 `speak` method
- **+1** attempt (calls `super.speak()` *and* method header matches abstract method, OK if `abstract` left in)
- **+1/2** correct value returned

Part C: Kennel - `allSpeak` **4 pts**

+1 loop over `petList`
- **+1/2** attempt
- **+1/2** correct (must access `petList`)

+1 1/2 get pet from `petList` (no deduction for missing downcast from `petList`)
- **+1/2** attempt
- **+1** correct (local variable must be type `Pet`)

+1 1/2 print `p.getName()` and `p.speak()` for pet `p` (local variable not necessary)
- **+1/2** attempt (must have `xxx.getName()` or `xxx.speak()`, for some xxx)
- **+1** correct

Note: if done in-line with no local, no deduction for missing downcast.

Usage:
- **-1/2** `public` instance variable
- **-1** parent class name instead of `super`
- **-1/2** `getName` is overridden (other than `super.getName`) in part (a) and/or part (b)

(No deduction for other additional methods or constructors.)

AP® Computer Science A
2004 SCORING GUIDELINES

Question 3

Part A: `numUnder` **3 pts**

+1 1/2 calculate total fish needed in the pond
- **+1/2** attempt (must have calculation involving `minDensity` and either `numRows` or `numCols`)
- **+1** correct (must round up; `double` is OK if fixed later)

+1 1/2 return number of fish to add to the pond
- (a) zero when no fish should be added
- (b) number of fish to be added when positive
- **+1/2** attempt –must correctly return (a) or (b) (OK without conditional), **or** attempt to return (a) or (b) with condition involving `minDensity`
- **+1/2** correct (OK even when total fish needed in pond is incorrect)
- **+1/2** type returned is correct (`int`)

Part B: `randomLocation` **3 pts**

+1 get instance of `Random`
- **+1/2** attempt (`Math.random` or `Random r = new Random` gets this 1/2 point)
- **+1/2** correct (uses `RandNumGenerator.getInstance()`)

+1 generate random row and column in correct range
- **+1/2** attempt (must use some aspects of Random and env/loc)
- **+1/2** correct (must have instance of `Random` or correct use of `Math.random`) (lose this 1/2 if erroneous check for empty location is included)

+1 create and return location
- **+1/2** attempt (uses a `Location` object)
- **+1/2** correct

Part C: `addFish` **3 pts**

+1 loop until correct `numToAdd` have been added
- **+1/2** attempt
- **+1/2** correct

+1 generate random location until empty one is found
- **+1/2** attempt (must call `randomLocation` within loop for adding fish, must attempt to check if empty)
- **+1/2** correct

+1 create fish at this location (note: no local `Fish` variable is needed)
- **+1/2** attempt (`Fish(loc)` or `generateChild(loc)` gets this 1/2)
- **+1/2** correct (lose this 1/2 for adding twice, e.g., `theEnv.add(new Fish(…))`)

AP® Computer Science A
2004 SCORING GUIDELINES

Question 4

Part A: `forwardMoveBlocked` **1 pt**

+1 return boolean
- **+1/2** check a dir/pos pair
- **+1/2** correct

Part B: `move` **5 pts**

+1 check for item(s) on current tile and remove one
- **+1/2** attempt on current tile (might try to remove all items)
- **+1/2** correct

+1 1/2 check required conditions in context of attempt to move/turn
(body of each check must refer to `pos` or `facingRight`)
- **+1** separate check for empty tile (e.g., not in ELSE)
- **+1/2** check `forwardMoveBlocked`

+1 change direction (set direction to some value relative to current direction)
- **+1/2** toggle value
- **+1/2** if and only if originally blocked

+1 1/2 move (set position to value(s) relative to current position)
- **+1/2** attempt 2 directions (change position, not value at position)
- **+1/2** move 1 tile in proper direction
- **+1/2** if and only if originally not blocked

Part C: `clearHall` **3 pts**

+1/2 declare and initialize counter (must have some extra context relevant to counting)

+1 loop until done
- **+1/2** call to `hallIsClear` in loop
- **+1/2** correct

+1 robot action (in context of a loop)
- **+1/2** call `move`
- **+1/2** correctly determine number of times `move` is called

+1/2 always return number of times `move` is called (no credit for returning 0 with no call to `move` in code)