AP® COMPUTER SCIENCE A
GENERAL SCORING GUIDELINES

Apply the question assessment rubric first, which always takes precedence. Penalty points can only be deducted in a part of the question that has earned credit via the question rubric. No part of a question (a, b , c) may have a negative point total. A given penalty can be assessed only once for a question, even if it occurs multiple times, or in multiple parts of that question. A maximum of 3 penalty points may be assessed per question.

1-Point Penalty

(w) Extraneous code that causes side effect (e.g. printing to output, incorrect precondition check)
(x) Local variables used but none declared
(y) Destruction of persistent data (e.g., changing value referenced by parameter)

Mr Lee's 1-Point Penalty:

- Inefficient, "long winded" or "messy" difficult to understand code which takes longer to write than standard more efficient solutions.
    - In an exam you need to save time by writing quickly hand writable efficient code which is easy for AP readers to understand.

No Penalty

- Extraneous code with no side effect (e.g., precondition check, no-op)
- Spelling/case discrepancies where there is no ambiguity*
- Local variable not declared provided other variables are declared in some part
- Keyword used as an identifier
- Common mathematical symbols used for operators (x • ÷ ≤ ≥< > ≠)
- `[ ]` vs. `()`
- Extraneous [ ] when referencing entire array
- `[i,j]` instead of `[i] [j]`
- = instead of == and vice versa
- Missing `{}` where indentation clearly conveys intent
- Missing `()` around `if` or `while` conditions

** Spelling and case discrepancies for identifiers fall under the "No Penalty" category only if the correction can be unambiguously inferred from context; for example, "total" instead of "totl". As a counterexample, that if the code declares "int G=99 , g=O; ", then uses "while (G < 10) " instead of "while (g < 10) ", the context does not allow for the reader to assume the use of the lower-case variable.*

2D Arrays – LightBoard FRQ

A `LightBoard` models a two-dimensional display of lights, where each light is either on or off, as represented by a `boolean` value. You will implement two code segments, one to initialize the display and another to evaluate a light.

```
/** The lights on the board, where true represents on and
 *  false represents off.
*/
boolean[][] lights;
```

(a) Write a code segment, which initializes `lights` so that each light is set to on with a 40% probability. The notation `lights[r][c]` represents the array element at row `r` and column `c`.

Complete the code segment below.

```
/** Initializes lights with numRows rows and
 *  numCols columns.
 *  Precondition: numRows > 0, numCols > 0
 *  Postcondition: each light has a 40% probability of
 *  being set to on.
 */
int numRows, numCols;
```

(b) Write a code segment which computes and prints the status of a light at a given row and column based on the following rules.

1. If the light is on, print the `boolean` value `false` if the number of lights in its column that are on is even, including the current light.
2. If the light is off, print the `boolean` value `true` if the number of lights in its column that are on is divisible by three.
3. Otherwise, print the light's current status.

For example, suppose that `numRows = 7` and `numCols = 5` and the code segment for part (a) above creates a light board with the initial state shown below, where `true` represents a light that is on and `false` represents a light that is off. Lights that are off are shaded.

`lights`

	0	1	2	3	4
0	true	true	false	true	true
1	true	false	false	true	false
2	true	false	false	true	true
3	true	false	false	false	true
4	true	false	false	false	true
5	true	true	false	true	true
6	false	false	false	false	false

Sample runs of the code segment you are being asked are shown below.

`int row, col;`	Value Printed	Explanation
`row = 0;` `col = 3;`	`false`	The light is on, and the number of lights that are on in its column is even.
`row = 6;` `col = 0;`	`true`	The light is off, and the number of lights that are on in its column is divisible by 3.
`row = 4;` `col = 1;`	`false`	Prints the light's current status.
`row = 5;` `col = 4;`	`true`	Prints the light's current status.

Information for this question

```
boolean[][] lights;
int numRows, numCols;
int row, col;
```

Complete the code segment below.

```
/** Evaluates a light in row index row and column index
 *  col and prints a status as described in part (b).
 *  Precondition: row and col are valid indexes in lights.
 */
int row, col;
```