2D Arrays – GrayImage FRQ

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 – GrayImage FRQ

A grayscale image is represented by a 2-dimensional rectangular array of pixels (picture elements). A pixel is an integer value that represents a shade of gray. In this question, pixel values can be in the range from 0 through 255, inclusive. A black pixel is represented by 0, and a white pixel is represented by 255.

```
final int BLACK = 0;
final int WHITE = 255;

/** The 2-dimensional representation of this image.
 *  Guaranteed not to be null.
 *  All values in the array are within the range [BLACK, WHITE],
 *  inclusive.
 */
int[][] pixelValues;
```

(a) Write a code segment that it prints the number of pixels in the image that contain the value `WHITE`. For example, assume that `pixelValues` contains the following image.

	0	1	2	3	4
0	**255**	184	178	84	129
1	84	**255**	**255**	130	84
2	78	**255**	0	0	78
3	84	130	**255**	130	84

Executing the code below would print 5 because there are 5 entries (shown in boldface) that have the value `WHITE`.

Complete the code segment below

```
/** @print the total number of white pixels in this image.
 * Postcondition: this image has not been changed.
 */
```

2D Arrays – GrayImage FRQ

(b) Write a code segment that modifies the image by changing the values in the array `pixelValues` according to the following description. The pixels in the image are processed one at a time in row-major order. Row-major order processes the first row in the array from left to right and then processes the second row from left to right, continuing until all rows are processed from left to right. The first index of `pixelValues` represents the row number, and the second index represents the column number.

The pixel value at position (`row, col`) is decreased by the value at position (`row + 2, col+ 2`) if such a position exists. If the result of the subtraction is less than the value `BLACK`, the pixel is assigned the value of `BLACK`. The values of the pixels for which there is no pixel at position (`row + 2, col + 2`) remain unchanged. You may assume that all the original values in the array are within the range `[BLACK, WHITE]`, inclusive.

The following diagram shows the contents of the array `pixelValues` before and after the code below is executed. The values shown in boldface represent the pixels that could be modified in a grayscale image with 4 rows and 5 columns.

Before execution of the code below

	0	1	2	3	4
0	**221**	**184**	**178**	84	135
1	**84**	**255**	**255**	130	84
2	78	255	0	0	78
3	84	130	255	130	84

After execution of the code below

	0	1	2	3	4
0	**221**	**184**	**100**	84	135
1	**0**	**125**	**171**	130	84
2	78	255	0	0	78
3	84	130	255	130	84

Information repeated from the beginning of the question

```
final int BLACK = 0;
final int WHITE = 255;
int[][] pixelValues;
```

Complete the code segment below

```
  /** Processes this image in row-major order and decreases the
   *  value of each pixel at position (row, col) by the value of
   *  the pixel at position (row+ 2, col+ 2) if it exists.
   *  Resulting values that would be less than BLACK are replaced
   *  by BLACK.
   *  Pixels for which there is no pixel at position
   *  (row + 2, col+ 2) are unchanged.
   */
```