2D Arrays – Successors 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 – Successors FRQ

This question involves reasoning about a two-dimensional (2D) array of integers. You will write a code segment to process a 2D integer array that contains consecutive values. Each of these integers may be in any position in the 2D integer array. For example, the following 2D integer array with 3 rows and 4 columns contains the
integers 5 through 16, inclusive.

2D `int` array

	0	1	2	3
0	15	5	9	10
1	12	16	11	6
2	14	8	13	7

Write a code segment that takes an `int` value `num` and an 2D `int` array `intArr` and prints the position of `num` in `intArr`. If `num` is not an element of `intArr`, the code segment prints `null`.

For example, assume that array `intArr` is as shown above.

- If `num = 8` the code segment should print the "`2, 1`" because the value *8* appears in `intArr` at row 2 and column 1.
- If `num = 17` the code segment should print `null` because the value `17` does not appear in `intArr`.

Complete the code segment below.

```
/** Prints the position of num in intArr;
 *  prints null if no such element exists in intArr.
 *  Precondition: intArr contains at least one row.
 */
int num = 8;
int[][] intArr = {{15, 5, 9, 10},
                  {12, 16, 11, 6},
                  {14, 8, 13, 7}};
```