1D Arrays – DivBySum 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.*

1D Arrays – DivBySum FRQ

A code segment is intended to print the sum of all the elements in the `int` array parameter `arr` that are divisible by the `int` parameter `num`. Consider the following examples, in which the array `arr` contains `{4, 1, 3, 6, 2, 9}`.

If `arr` is as above and `num = 3`, the code segment should print `18`, which is the sum of `3`, `6`, and `9`, since those are the only integers in `arr` that are divisible by `3`.
If `arr` is as above and `num = 5`, the code segment should print `0`, since none of the integers in arr are divisible by 5.

Complete the code segment using an enhanced `for` loop. The method must use an enhanced `for` loop to earn full credit.

```
/** Prints the sum of all integers in arr that are
 *  divisible by num
 *  Precondition: num > 0
 */
int[] arr = {4, 1, 3, 6, 2, 9};
int num = 3;
```