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 – NumberCube FRQ

A statistician is studying sequences of numbers obtained by repeatedly tossing a six-sided number cube. On each side of the number cube is a single number in the range of 1 to 6, inclusive, and no number is repeated on the cube. The statistician is particularly interested in runs of numbers. A run occurs when two or more consecutive tosses of the cube produce the same value. For example, in the following sequence of cube tosses, there are runs starting at positions 1, 6, 12, and 14.

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Result	1	5	5	4	3	1	2	2	2	2	6	1	3	3	5	5	5	5

You will implement a code segment that collects the results of several simulated tosses of a number cube and another code segment that calculates the longest run found in a sequence of tosses.

(a) Write a code segment that takes a number of tosses as a parameter. The code segment should print an array of the values produced by simulating the tossing of a six-sided number cube the given number of times.

Complete the code segment below.

```
/** Creates an array of the values obtained by
 *  simulating the tossing of a number cube
 *  numTosses times.
 *  @param numTosses the number of tosses to be recorded
 *         Precondition: numTosses > 0
 *         Precondition: values.length > 0
 *  @prints an array of numTosses values
 */
int numTosses;
```

(b) Write a code segment that takes as its parameter an array of integer values representing a series of number cube tosses. The code segment prints the starting index in the array of a run of maximum size. A run is defined as the repeated occurrence of the same value in two or more consecutive positions in the array.

For example, the following array contains two runs of length 4, one starting at index 6 and another starting at index 14. The code segment may print either of those starting indexes.

If there are no runs of any value, the code segment prints -1.

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Result	1	5	5	4	3	1	2	2	2	2	6	1	3	3	5	5	5	5

Complete the code segment below.

```
/** Prints the starting index of a longest run of two or
 *  more consecutive repeated values in the array values.
 *  @param values an array of integer values representing a series of number
 *                cube tosses
 *         Precondition: values.length > 0
 *  @print the starting index of a run of maximum size;
 *         -1 if there is no run
 *  Assume that this code segment is written after
 *  correctly working code for part (a).
 */
int[] values;
```