2D Arrays – RouteCipher 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 – RouteCipher FRQ

A route cipher encrypts (puts into a coded form) a message by changing the order of the characters in the message. The route cipher in this question fills a two-dimensional array with single-character substrings of the original message in row-major order.

For example, the word `"Surprise"` can be encrypted using a 2-row, 4-column array as follows.

Original Message: `"Surprise"` →

Contents of Array

"S"	"u"	"r"	"p"
"r"	"i"	"s"	"e"

```
// A two-dimensional array of single-character strings.
String[][] letterBlock;

// The number of rows of letterBlock
int numRows;

// The number of columns of letterBlock
int numCols;

String str;
```

Write a code segment that fills the two-dimensional array `letterBlock` with one-character strings from the `String str`.

The array must be filled in row-major order -- the first row is filled from left to right, then the second row is filled from left to right, and so on, until all rows are filled.

If the length `str` is smaller than the number of elements of the array, the `String "A"` is placed in each of the unfilled cells. If the length of `str` is larger than the number of elements in the array, the trailing characters are ignored.

For example, if `letterBlock` has 3 rows and 5 columns and `str` is the string `"Meet at noon"`, the resulting contents of `letterBlock` would be as shown in the following table.

"M"	"e"	"e"	"t"	" "
"a"	"t"	" "	"n"	"o"
"o"	"n"	"A"	"A"	"A"

2D Arrays – RouteCipher FRQ

If `letterBlock` has 3 rows and 5 columns and `str` is the string `"Meet at midnight"`, the resulting contents of `letterBlock` would be as shown in the following table.

"M"	"e"	"e"	"t"	" "
"a"	"t"	" "	"m"	"i"
"d"	"n"	"i"	"g"	"h"

The following expression may be used to obtain a single-character string at position k of the string `str`.

```
    str.substring(k, k + 1)
```

Complete the code segment below.

```
  /** Places a string into letterBlock in row-major order.
   *  str the string to be processed
   *  Postcondition:
   *    if str.length() < numRows * numCols, "A" is placed
   *    in each unfilled cell
   *    if str.length () > numRows * numCols, trailing
   *    characters are ignored
   */
```