**APCS 2000 Exam, Java Version A4 Free Response**

one way of encrypting a word is to encrypt pairs of letters in the word together. A scheme to do this is to fill a 6 x 6 square with the 26 capital letters of the alphabet and then digits '0' through '9'. Each letter and digit appears exactly once in the square.

To encrypt a letter pair, the rectangle formed by the two letters is used. Each letter of the original pair is replaced by the letter located on the same row and in the other corner of the rectangle. If both letters happen to be in the same row or column, the letters are swapped..

For example, in the following arrangement AP is encrypted as DM.

S T U V W X

Y Z 0 1 2 3

4 5 6 7 8 9

A---B---C---D E F

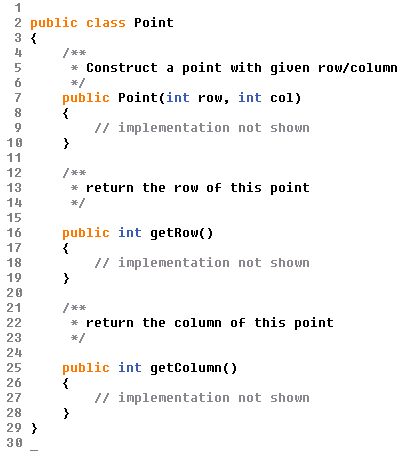
| |

G H I J K L

| |

M---N---O---P Q R

Consider the following declaration for a class that uses this scheme to encrypt a word . Only the method/constructors headers are shown, not the bodies.



The class Encryptor is partially shown below.



**Part A**

Write member function getCoordinates, as started below. getCoordinates takes a single character string representing a capital letter or a digit and returns its row and column in the 2-dimensional array myMatrix.

The following example shows the point locations of the String ch in the given matrix.

|  |  |  |
| --- | --- | --- |
| **myMatrix** | **ch** | **Point coordinates** |
| S T U V W X  Y Z 0 1 2 3  4 5 6 7 8 9  A B C D E F  G H I J K L  M N O P Q R | p  8  M | row = 5 col = 3  row = 2 col = 4  row = 5 col = 0 |

Complete method getCoordinates below.

/\*\*

\* find/return a pair for the single character string ch

\* precondition: ch is a single character string with

\* "A" <= ch <= "Z" or "0" <= ch <= "9"

\* @return the coordinates of ch in the encryption matrix

\*/

private Point getCoordinates(String ch)

**Part B**

Write method encryptTwo as started below. Method encryptTwo is passed a two-character string and returns an encoded two-character string.

The encoding of a letter pair is formed as follows.

* If both letters are in the same row or column, swap the two letters.
* Otherwise, find the other two corners of the rectangle formed by the two letters. Each letter of the original pair is replaced by the letter located on the same row and in the other corner of the rectangle.

For example, to encrypt the string "NE", look at the rectangle with corners "N" and "E". The encrypted letter pair is "QB" because "Q" is the letter at the other corner on the same row as "N", and "B" is the letter at the other corner on the same row as "E".

S T U V W X

Y Z 0 1 2 3

4 5 6 7 8 9

A B---C---D---E F

| |

G H I J K L

| |

M N---O---P---Q R

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Letters:** | "BR" | "NE" | "ET" | "RE" | "TH" | "PR" | "GG" |
| **Encrypted:** | "FN" | "QB" | "BW" | "QF" | "HT" | "RP" | "GG" |
|  |  |  |  |  |  |  |  |

In writing encryptTwo you may call getCoordinates specified in part (a). Assume that getCoordinates works as specified, regardless of what you wrote in part (a).

Complete method encryptTwo below.

/\*\*

\* encrypt a two-character string

\* @precondition pair.length() == 2; pair has two characters

\* @param pair is the string encrypted

\* @return an encrypted form of pair

\*/

private String encryptTwo(String pair)

**Part C**

Write method encryptWord, as started below. Method encryptWord takes a word parameter and returns a string that contains the encryption of that word. Every two letters of the word are examined and encrypted by replacing the original letters with those located in the opposite corners of the rectangle formed by the two letters. If the original word contains an odd number of letters the last letter is unchanged.

The following are examples of encrypted words using the 2-dimensional array shown below.

S T U V W X

Y Z 0 1 2 3

4 5 6 7 8 9

A B C D E F

G H I J K L

M N O P Q R

|  |  |  |  |
| --- | --- | --- | --- |
| **Word:** | "COMPUTER" | "SCIENCE" | "STUDENTS" |
| **Encrypted:** | "OCPMTUFQ" | "UAKCOBE" | "TSVCBQST" |

In writing encryptWord you may call encryptTwo specified in part (b). Assume that encryptTwo works as specified, regardless of what you wrote in part (b).

/\*\*

\* Converts a string to an encrypted form.

\* @precondition: word contains only letters A-Z or

\* digits 0-9

\* @param word to be encrypted

\* @return the encrypted form of word

\*/

public String encryptWord(String word)