4. 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 the ten 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.

```
struct Point
{
  int row;
  int col;

  Point();                          // default constructor
  Point(int newRow, int newCol); // sets row to newRow, col to newCol
};

class Encryptor
{
  public:
    Encryptor();
       // fills the matrix with the 26 letters of the alphabet
       // and the 10 digits '0' through '9'

    apstring EncryptWord(const apstring & word) const;
       // returns an encrypted form of the word

  private:
    apmatrix<char> myMat;

    apstring EncryptTwo(const apstring & pair) const;
       // returns an encrypted form of the pair

    Point GetCoordinates(char ch) const;
       // returns the coordinates of ch in the 2-dimensional array
};
```

(a) Write member function `GetCoordinates`, as started below. `GetCoordinates` takes a given letter or digit and returns its row and column in the 2-dimensional array. Assume that the parameter `ch` is a capital letter in the range `'A'` through `'Z'` or a digit in the range `'0'` through `'9'`.

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

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

Complete function `GetCoordinates` below.

```
Point Encryptor::GetCoordinates(char ch) const
// precondition:  'A' ≤ ch ≤ 'Z' or '0' ≤ ch ≤ '9'
// postcondition: returns the row and column number of the
//                location of ch in myMat
```

(b) Write member function `EncryptTwo`, as started below. `EncryptTwo` is passed a two-character string and returns an encoded two-character string.

The encoding of a letter pair is formed as follows.

1. If both letters are in the same row or column, swap the two letters.
2. 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 a letter pair, say `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 function `EncryptTwo` below.

```
apstring Encryptor::EncryptTwo(const apstring & pair) const
// precondition:  pair.length() is 2
// postcondition: returns an encoded two-character string
```

(c) Write member function `EncryptWord`, as started below. `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).

Complete function `EncryptWord` below.

```
apstring Encryptor::EncryptWord(const apstring & word) const
// precondition:  word contains only capital letters 'A' through 'Z'
//                and digits '0' through '9'.
// postcondition: returns an encrypted version of word, in which every
//                two letters have been 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 left unchanged.
```

END OF EXAMINATION