CS 105 – Lab #9 – Vigenère cipher

You have already learned in class about the theory of the Vignere cipher. Now it's time to implement it in Pascal.

Log into the computer as usual. It's a good idea to have two terminal windows open so that you can see them side by side on the screen. Create a new folder called lab9 to hold today's work. Make sure that your current working directory is lab9 before continuing. In other words, when you type the `pwd` command, the system should tell you that you are inside lab9.

To recap, the Vigenère cipher essentially combines multiple Caesar ciphers into one. You start with two inputs: a plaintext message, and a code word (key). Corresponding letters from the plaintext and key are added. Often, the message is longer than the key, so that we need to wrap around and start the key over at the beginning. Here is an example: Suppose the message is ATTACKATDAWN and the key is LEMON. Once we get to the 6th letter of the plaintext message (as well as the 11th, 16th, …), we have to start LEMON over at the L. Here is what we add:

```
Plain:  ATTACKATDAWN
Key:    LEMONLEMONLE
```

We treat each letter A-Z as a number 1-26 when we add. And if the sum exceeds 26, we subtract 26 so that the result is another letter. Thus, the first five enciphered letters are:

A + L = M
T + E = Y
T + M = G
A + O = P
C + N = Q

1. Let's practice an example. Encipher the phrase "fish stew", using the key word "table". Use only capital letters, and omit all spaces.

Here is the idea behind our program:

- ASCII code assigns A the value 65. But it would be nice if we could number the letters starting A at 1. So, we use a constant offset of 64 to subtract from the ASCII.
- Your program will first ask for a message, and a key, putting each into uppercase.
- Now that we have a plaintext message and key, we iterate through each letter in the message.
    - Get the proper key letter to correspond to the message letter (wrap around if necessary).
    - Convert a letter from the message and a letter from the key into numbers 1-26.
    - Add the plain letter and key letter. Adjust the sum if necessary to ensure it is 1-26.
    - Convert the number back to the letter.

Use nano to create a new program called vig.pas. Here is the source code. You will need to supply some of the details in the implementation. They are numbered 1 through 4 in the code below.

```
(* vig.pas - Vigenere cipher.  Let's assume it's just looking at capital
 * letters.  Add corresponding letters of plaintext and key to produce the
 * ciphertext letters.  Convert to upcase first.
 * The constant asciioffset allows us to treat A as 1 instead of 65.
 * Assume the message and key have only letters.  No spaces, punctuation.
 *)
program vig;

const
  asciioffset = 64;

var
  inputtext, outputtext, key : string;
  inputchar, outputchar, keychar : char;
  inputindex, keyindex, inputvalue, keyvalue, outputvalue : integer;

begin
  writeln('This program will encrypt using the Vigenere cipher.');
  write('Enter plaintext:  ');
  readln(inputtext);
  write('Enter the key:    ');
  readln(key);

  (* 1.  Convert both the inputtext and key to their upcase versions. *)
```

```
  (* We need to keep track of our position in both the plaintext and key.
   * Note that once we get to the end of the key, we must start at beginning.
   *)
  keyindex := 1;
  outputtext := '';

  (* 2.  Loop for each letter in the inputtext string.  Fill in the blank: *)
  for inputindex := 1 to _____________ do
    begin
      inputchar := inputtext[inputindex];
      keychar := key[keyindex];

      inputvalue := ord(inputchar) - asciioffset;
      keyvalue := ord(keychar) - asciioffset;
      outputvalue := inputvalue + keyvalue;

      (* 3.  If outputvalue is greater than 26, then subtract 26 from it. *)



      outputvalue := outputvalue + asciioffset;
      outputchar := chr(outputvalue);
      outputtext := outputtext + outputchar;

      (* Having consumed a key letter, move to next one. *)
      keyindex := keyindex + 1;

      (* 4.  Add 1 to the keyindex.  After doing so, if the keyindex is
       * greater than the length of the key, reset it to 1.
       *)




    end;

  writeln();
  writeln('Plaintext:  ', inputtext);
  writeln('Key:        ', key);
  writeln('Ciphertext: ', outputtext);
end.
```

2. Run your program using plaintext FURMANPALADINS and key PURPLE. What is the program's output?

3. Let's think about what is happening while the program just ran with your example input.
   a. Which letter of the key is located at keyindex = 2?

   b. Which inputtext letters are added to this key letter?

   c. What is the inputindex corresponding to those plaintext letters?

Finally, let's create a decryption program. Copy vig.pas into vig2.pas by entering this terminal command: `cp vig.pas vig2.pas`. Then, use nano to edit vig2.pas. You need to make the following changes:

- Change the comment and program declaration.
- Change the input prompt so that it asks for ciphertext instead of plaintext.
- At the end of the program, change the output introductions so that the plaintext and ciphertext are correctly identified.
- Inside the for-loop, change the way that the outputvalue is assigned. Subtract instead of add. And if the resulting output value is too low, add 26.

Because the program refers to the secret message generically as inputtext and outputtext instead of ciphertext and plaintext, we do not need to change the names of these variables.

4. Let's verify that your vig.pas and vig2.pas both work correctly. √

5. How did you modify the if-statement in the for-loop to check to see if the outputvalue was too low? What is the lowest acceptable outputvalue?

6. What would you say are the weaknesses of the Vigenère cipher?