GG250 Lab 5
Some cryptology

Background: During World War II, the Germans developed a very sophisticated, mechanical encryption machine capable of encoding and decoding messages in the field. The idea was that if the Allies intercepted the messages they would not be able to unscramble them and learn of the German plans. Polish mathematicians broke the code and tipped off the British just before the war started. The British in turn built a mechanical computer used to aid in the deciphering of the coded messages – it was a turning point in the war. The German code machine was called Enigma[1].

We will make our own, much simpler coding machine in Matlab. The essence of the coding machine is to replace each letter in a text message with another letter in a systematic way so that it is possible to reverse the process and recover the original message. The circular cut-out keys can be used to mimic the process. Our Matlab product will be a function whose first line is

```
function code = scramble_name (text, scheme, key, mode)
```

Again, substitute your name for “name”. The input parameters are:

`text`	A text message (a sentence) in single quotes.
`scheme`	An integer (1 or 2) to set the kind of algorithm to use.
`key`	Positive integer offset used in the encoding and decoding
`mode`	Use +1 for encoding and -1 for decoding

Your function must test the validity of the last 3 arguments. The output parameter is

`code`	The encoded text message.

We will allow for two slightly different schemes:

1. The Caesar code, one of the oldest codes around. Replace each letter with one that is *n* steps further down the alphabet, i.e., for *n* = 4 A becomes E, B becomes F, etc. This offset *n* is the **`key`** argument above. We want to apply the scheme only to the letters in the message. If adding *n* takes you outside the letter range you must ‘wrap around’ to the beginning (i.e., V becomes Z, W becomes A, X becomes B). You can simulate this process by assembling your CIA decoding device and rotating the inner disc so the number *n* appears in the box, then find the letters you want to encode on the outside circle and read off the juxtaposed encoded letters on the inner circle.
2. The rotation code. Similar to the Caesar code, we shift each letter in the message by *n*, but we also increment *n* by 1 after each step. I.e., for *n* = 4 A becomes E, B becomes G, C becomes I, etc. Use your discs to simulate this

[1] For more information, watch the movie “Enigma” available at the Sinclair AV Center.

encoding too, but after the initial setup you must rotate the disc by one step after encoding each character.

In both cases, you just reverse your procedure to decode an encrypted message. When simulating this process in your Matlab code, you will want to take the ASCII values for A–Z (65-90), subtract the ASCII value for A and get numbers in the 0 – 25 range. Now add the offset and use modulo 26 to make sure the result is in the 0 – 25 range. Finally add back 65 to get ASCII values. Do something similar for lower case letters and use the char function to return a proper text string. Anything that is not a letter should be left unchanged.

Assignments:

1. In a World document, prepare
   a. Pseudo-code that outlines what your function should do
   b. Flowchart that illustrates the program flow for your function.

   Attach this 2-page Word file to your email submission.
2. Develop a fully documented function scramble_name.m that follows the specifications given above. At a minimum, you will need to use a few if-tests and the functions *find*, *mod*, and *char*. During development, test it on simple strings such as in these examples:

   code= scramble_name (‘ABCDEFGHIJKLMNOPQRSTUVWXYZ’, 1, 5,+1);
   code = scramble_name (‘abcdefghijklmnopqrstuvwxyz’, 2, 5, +1);

   scramble_name (code, mode, key, -1) should give you back your original message as long as your are using the same mode and key. Make sure you test that encoding followed by decoding yields the original message. E.g. code = scramble_name (‘The Duck of Death’, 2, 8, +1) gives code = ‘Bqo Phqz fx Xzwqf’ and scramble_name (‘Bqo Phqz fx Xzwqf’, 2, 8, -1) returns ‘The Duck of Death’.
3. Submit your Matlab function as an attachment. Use it to determine the original messages, schemes, and keys used to produce these two codes:

   Bnhb nzqf ixrl vppqmm eq?
   Lxy hksetldjbq: Bvwehq kxxgal xd teixg XM7 cw rokvrqsf

   Give the answers in the body of the email and submit it to gg250-lab@hawaii.edu.