1. Caesar’s Cipher

A simple example of conventional cryptography is the substitution cipher. A substitution cipher substitutes one piece of information for another, often by offsetting letters of the alphabet. One example is Julius Caesar's cipher, which offsets the alphabet by a certain number of characters. The number of characters offset is called the key.

For example, if we encode a message using Caesar's key value of 3, we offset the alphabet so that the 4th character (D) begins the alphabet while the first three characters (A, B, C) are “wrapped around” to the end. Therefore, starting with the original alphabet

\[
\text{ABCDEFGHIJKLMNOPQRSTUVWXYZ}
\]

and offsetting by 3 characters, you get

\[
\text{DEFGHIJKLMNOPQRSTUVWXYZABC}
\]

where D represents A, E represents B, F represents C, and so on.

Using this scheme, the plaintext "SECRET" encrypts as "VHFUHW". To allow someone else to read the ciphertext, you tell him or her that the key is 3.

In this project, we will consider a modified version of Caesar’s Cipher where we not only offset the alphabets, but shift the position of the characters themselves.

For instance, if we had a “key” of 3 and a “phase shift” of 4, then to encode “SECRET” we first convert it to “VHFUHW” as above, and then we place the list of characters starting from the 5th position at the beginning of the string, with the first 1 to 4 characters being appended to the end of the string. In this example, the list of characters starting at the 5th position is “HW”, and we append the first 4 characters (“VHFU”) to the end, to get HWVHFU.

Note that your input strings will contain numeric characters, spaces, and punctuation marks. These characters should not be converted by the “key” into any other characters, but they should be phase shifted. I.e., a “comma” in the source string will still be a “comma” in the encrypted string, but it may or may not be in the same position in the string depending on the value of “phase shift.” Thus, we only change the identities of alphabetical characters (both lower and upper case) in our algorithm.
The input string for encryption may be in upper or lower case, but you may just output the decrypted string in upper case in this project. Similarly, the input string for decryption may be in either case, and you can just output the original text in upper case.

**More examples:**
Source string: The Quick Brown fox
Key: 2
Phase shift: 1
Intermediate result after using key: VJG SWKEM DTQYP HQZ
After Phase shifting: JG SWKEM DTQYP HQZV
*Note in the example above that the output is in uppercase, and that the space character between words is not changed into any other character.*

Source string: 4 ducks in 5th avenue
Key: 1
Phase shift: 4
Intermediate result after using key: 4 EVDLT JO 5UI BWFOVF
After phase shifting: DLT JO 5UI BWFOVF4 EV
*Note in this example that the numeric characters ‘4’ and ‘5’ were not changed into any other characters, but their positions in the encrypted string are different from their positions in the source string.*

**To do:** Write 2 MATLAB functions with the following specification:

```matlab
function secret = encrypt(sourceStr, key, phaseShift)
% Encrypt and return source string (sourceStr) using modified Caeser’s cipher, % as described in P4. Output is in uppercase.

function plaintxt = decrypt(secret, key, phaseShift)
% Decrypt and return secret string according to modified Caeser’s cipher, % as described in P4. Output is in uppercase.
```

Also write a MATLAB script called `part1.m` that outputs in a nicely formatted fashion the encrypted strings for:

- “640K (RAM) ought to be enough for anybody. - Bill Gates, 1981”
  where **key** is 3 and **phase shift** is 3. The quotes at the start and end of the string are for clarity and are *not* part of the string to encrypt.

- “BREAKFAST.COM halted!!!!!!...Cereal Port Not Responding. : )”
  where **key** is 7 and **phase shift** is 10.

The script should also call your decrypt function to decrypt the following string and print out the answer in a nicely formatted fashion:

- “ b dpme xjoufs!Ju xbt”
  where **key** is 1 and **phase shift** is 6. Note that there is a space (‘ ‘) before the character b.

**Note:** Do not use MATLAB’s predefined functions `strfind`, `strrep`, `strmatch`, `strtok`. 2
Get both your \texttt{encrypt} and \texttt{decrypt} functions to return output that match the case of the input string. That is, if position $i$ in the input string is a lower case character, then position $i$ in the intermediate output string \textit{before phase shift} should be lower case too, and vice versa. If you decide to do this, the first comment at the start of the two functions should read:

\begin{verbatim}
    % Bonus implemented--match case
\end{verbatim}

Note that if you attempt the bonus, but inadvertently introduce bugs into the base functionality of your programs, core points (not just bonus) will be deducted accordingly.