Specification for Assignment 7 of 8

Your submission ***must include a completed copy of the assignment header*** *that* ***must include your name*** *(as it appears on cuLearn),* ***student number****, and* ***section****.*

Your submission ***must be a source file named 'comp1405_f17_#########_a7.py'*** *(with the number signs replaced by your nine digit student number),* ***must be written*** *using the* ***Python 3*** *language, and* ***must run on the official virtual machine****.*

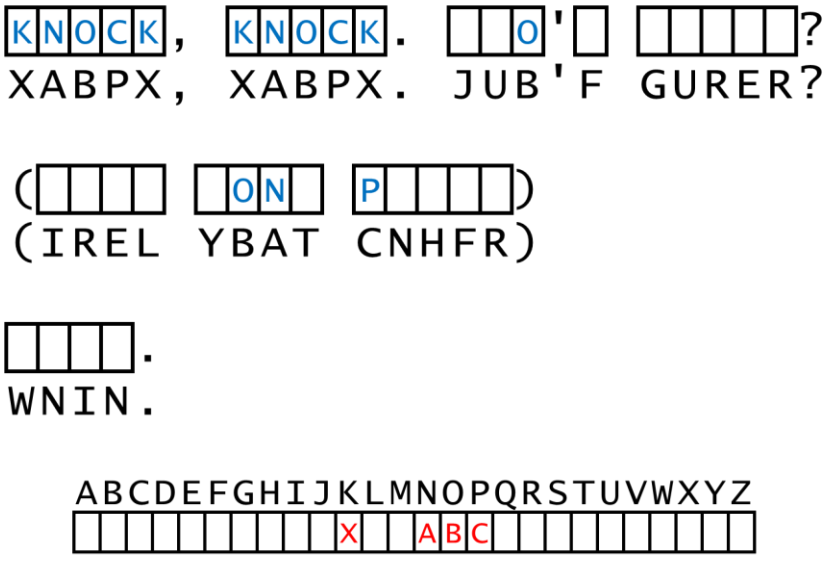
Do not compress your submission *into a "zip" file.*

Late assignments will not be accepted *and will* ***receive a mark of 0****.*

Submissions that crash *(i.e., terminate with an error) will* ***receive a mark of 0****.*

The due date for this assignment is Thursday, November 16, 2017, by 11:30pm.

For this assignment, you will create a program that allows users to work with cryptograms.



Your program must be **completely organized into a collection of functions** described by this document, and your program is **not permitted to use any global variables** whatsoever.

To accomplish this, **you will need to pass elements of data back and forth between functions using arguments and return values**. Although you will be provided some code for loading and saving text files, for every other function you will be informed exactly what the arguments and return values must be.

Specification for Assignment 7 of 8

Start by creating a new file and then, once you have inserted the header with your name and student number, add the following functions to your program.

```
'''
This function will load a text file.

@params             file_name, the name of the file to be loaded
@return             an uppercase string containing the data read from the file
'''
def load_text(file_name):
       file_hndl = open(file_name, "r")
       file_data = file_hndl.read()
       file_hndl.close()
       return file_data.upper()

'''
This function will save data to a text file.

@params             file_name, the name of the file to be saved
                    file_data, the data to be written to the file
@return             none
'''
def save_text(file_name, file_data):
       file_hndl = open(file_name, "w")
       file_hndl.write(file_data)
       file_hndl.close()
```

You will notice that both of these functions are preceded by a multi-line comment that has been wrapped in triple single quotation marks. **You are required to write a comment like this for each of the functions you create for this assignment**. This comment must begin with a short description of what the function does, contain a list (labeled @params) of indented items corresponding to the parameter variables you use to hold your function arguments, and contain a description of any return values (labeled @return). If the function you are writing has no params or returns, then you must still include the @params or @return tag and write (and indent) the word none.

Failure to observe this requirement will result in a penalty.

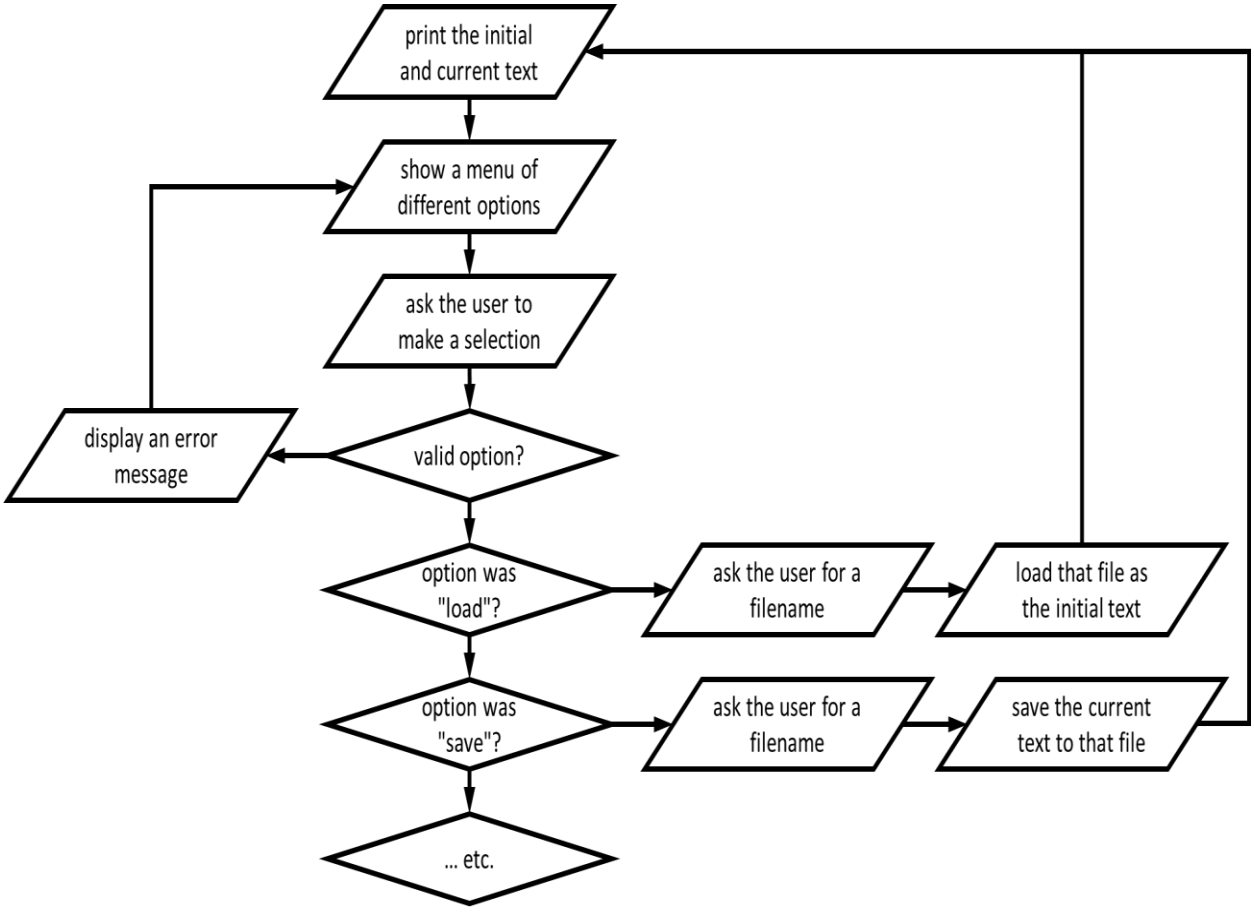
You must write a main function that takes no arguments and produces no return values. Additionally, as the very last line of your program, you must call this main function.

```
'''
This is the main function, responsible for the user interface.

@params                     none
@return                     none
'''
def main():

       # much of your code will go here


# this is the only line in your code that isn't inside a function definition
main()
```

Most of your program will be written into the function definition of that main function. This function is expected to provide a basic interface to the user. Your user should be able to **use the menu to select an option** and your program **will call the corresponding function (and input if necessary).** This will allow your user to load/save files and generate custom alphabets for encryption or decryption.

Please note that in order for you to be able to complete all of the functions specified on the following page, **your main function must have at least three variables**, for storing 1) **the "initial" unmodified text**, 2) **the "current" text being modified** by the user, and 3) **the current alphabet** being used for encoding and decoding.

A flowchart explaining the basic functionality expected is included below:



Every time your user sees the main menu you must display for them:

a) the "initial" unmodified text last loaded by the load function (initially empty)
b) the "current" text that your program most recently created by encoding or decoding
c) a list of options from which the user will select their next step

On the following page is the list of functions that you must implement.

You must **write a function called encode**. It must receive **two string arguments** and produce **one string return value**. The first argument will be the text to be encoded, and the second argument will the be alphabet to be used for this encoding. The return value will be the encoded text. Consider the following example for clarification:

```
a = "COMPUTERS MAKE VERY FAST, VERY ACCURATE MISTAKES."
b = "FGHIJKLMNOPQRSTUVWXYZABCDE"
c = encode(a, b)
# c should have a value of "HTRUZYJWX RFPJ AJWD KFXY, AJWD FHHZWFYJ RNXYFPJX."
```

You must **write a function called decode**. It must receive **two string arguments** and produce **one string return value**. The first argument will be the text to be decoded, and the second argument will the be alphabet to be used for this decoding. The return value will be the decoded text. Consider the following example for clarification:

```
a = "HTRUZYJWX RFPJ AJWD KFXY, AJWD FHHZWFYJ RNXYFPJX."
b = "FGHIJKLMNOPQRSTUVWXYZABCDE"
c = decode(a, b)
# c should have a value of "COMPUTERS MAKE VERY FAST, VERY ACCURATE MISTAKES."
```

You must **write a function called caesar_cipher_alphabet**. It must receive **one integer argument** and return **one string return value**. The argument is the amount by which the alphabet should be "shifted" to create a new alphabet for encoding or decoding. Read more about this cipher at https://learncryptography.com/classical-encryption/caesar-cipher. The return value will be the alphabet created. Consider the following example for clarification:

```
a = caesar_cipher_alphabet(3)
# a should have a value of "DEFGHIJKLMNOPQRSTUVWXYZABC"
```

~~You must **write a function called vigenere_cipher_alphabet**. It must **receive one string argument** and return **one string return value**. The argument is the keyword used to create a new alphabet for encoding or decoding. (The Vigenère cipher moves all the letters from a keyword to the beginning of the alphabet without changing their order). Read more about this cipher at https://learncryptography.com/classical-encryption/vigenere-cipher. The return value will be the alphabet created. Consider the following example for clarification:~~

```
a = vigenere_cipher_alphabet("HOWDY")
# a should have a value of "HOWDYABCEFGIJKLMNPQRSTUVXZ"
```

You must **write a function called cryptogram_alphabet**. It must receive **no arguments** and produce **one string return value**. This function will use input to ask the user to type an alphabet (i.e., type the 26 letters of the English alphabet in any order). If the argument contains all 26 letters then the argument, converted to uppercase if necessary, is the return value. If, on the other hand, the argument does not contain all 26 letters or contains duplicate letters, you will print an error message and return the unmodified alphabet "ABCDEFGHIJKLMNOPQRSTUVWXYZ".

Specification for Assignment 7 of 8

Your solution **must not import any libraries**, but you are **free to use** any of the following functions from the standard library: **list, range, upper, str, input, open, read, close, ord, chr, print, len**, and also **int, insert**, **append**, and **pop**.

Your program **must not use any of the string processing functions**. This includes, but is not limited to, find/rfind, index/rindex, replace, sort, strip, split, translate or the built-in encode or decode functions.

Your solution **may use the indexing operator** (i.e., you can use a[x] to get the element of a list or string that is located at index x) but your solution **must not use the slicing operator** (i.e., you cannot use a[x:y] to access the elements of the list or string that have indices between x and y-1).

Your solution **must not use list comprehensions** in any way.

Your solution **must not use dictionaries or maps** in any way either.

Please remember that **this assignment must be completed individually**. **Do NOT discuss this assignment with anyone** except the instructor or the teaching assistants, and **do NOT look for code samples on the internet** or from any other source.

As with the previous assignment, **you must create flowcharts** before you start writing your code for this assignment. You are expected to create a flowchart for each of the functions you will be writing. You will not be submitting any flowcharts on cuLearn, but if you visit the instructor or a teaching assistant **you will be asked to show your flowcharts before you will receive any assistance**.

You must **use meaningful variable names** and you must **comment your code**.

*** CORRECTION ***

You must **write a function called keyword_cipher_alphabet**. It must **receive one string argument** and return **one string return value**. The argument is the keyword used to create a new alphabet for encoding or decoding. (The keyword cipher moves all the letters from a keyword to the beginning of the alphabet without changing their order). Read more about this cipher at https://en.wikipedia.org/wiki/Keyword_cipher. The return value will be the alphabet created. Consider the following example for clarification:

```
a = keyword_cipher_alphabet("HOWDY")
# a should have a value of "HOWDYABCEFGIJKLMNPQRSTUVXZ"
```

To further clarify how to construct the keyword cipher alphabet, consider the sample keyword "ROBERT". This keyword contains the letters R, O, B, E, and T. (Please note that, although the keyword is six letters long, there are only five unique letters in the word "ROBERT". Thus, the cipher alphabet you create with this keyword must start with the letters R, O, B, E, and T, in that order.

To this, to create an alphabet that is exactly 26 letters long, you must add every letter of the alphabet that was not one of the letters in the keyword. Thus, since the keyword contains only the letters R, O, B, E, and T, you would first add A, then C, then D, then F, etc. (n.b., since B and E were already part of the cipher alphabet because they were letters of the keyword, we did not add them a second time).

The final cipher alphabet that should be produced (if your function received "ROBERT" as an argument) would be "ROBETACDFGHIJKLMNPQSUVWXYZ".