Homework #2 Data Structures Due: Sept. 16 (Saturday at 11:59 PM)

Objects: Practice designing a program and "reviewing" Python file usage. (NOTE: be sure to review the **example programs found** at: www.cs.uni.edu/~fienup/cs1520f17/homework/example_programs_hw2.zip)

File Encryption/Decryption Program: You are to **design** and write a program that allows a user to interactively select between encrypting or decrypting files:

- encrypting a user-specified text-file (`.txt` extension) using the encryption scheme described below to generate a new encoded file with the same name, except with a `.zzz` extension.
- decrypting a user-specified encrypted file (`.zzz` extension) using a user-specified keyword to generate a new text-file with the same name, except with a `.txt` extension.

The encryption scheme is a form of substitution cipher based on a user-specified positive integer key and the follow sequence of 78 characters: (Note: blank-space character between the "K" and "l")

```
                aA0bB1cC2dD3eE4fF5gG6hH7iI8jJ9kK lL,mM.nN?oO/pP;qQ:rR'sS"tT!uU@vV$wW%xX&yY-zZ=
sequece position: 012345678901234567890123456789012345678901234567890123456789012345678901234567
                0         1         2         3         4         5         6         7
```

The integer key is used to encrypt **only the first letter** in the message by "shifting" down the above sequence by that amount. For example, an integer key of 5 shifts 'B' to the encrypted 'd'. Shifting past the right end of the sequence wraps back to the beginning of the sequence. For example, shifting 5 from 'Z' encrypts as 'b'.

The sequence position of the first letter in the message is used as the shift amount for the second letter in the message. If the first letter was a 'B,' then 4 is used to shift the second letter in the message since 'B' is at position 4 (start with 'a' at position 0, 'A' at position 1, etc.). The sequence position of the second letter in the message is used as the shift amount for the third letter, etc. Any message character not in the above sequence (e.g., '<', '{') is copied to the encrypted message without modification with the previous shift amount carrying over to the next letter in the message to be encrypted. Consider the following example with key 5:

Four Line Message to Encrypt with Key = 5 in file message.txt	**Encrypted Message in file message.zzz**

`Be by the Union at 6 PM!`
 	
`<We'll study for Data Structures...>`
`- Sam`
	`dF/,z83al/fHvwb& t3RRa1g`
	
	`<qA$2w$2lnxb8;tf1oDtt dLklwvnlvw4ZZ>`
	`LJdSm`

Your program also needs to: (See www.cs.uni.edu/~fienup/cs1520f17/homeworks/example_programs_hw2.zip)

- validate a correct selection whether to encrypt, decrypt, or exit.
- validate a correct positive integer for the key.
- validate that the user specified file to encrypt/decrypt exists and force the user to reenter until they specify an existing file. Feel free to make the program more user-friendly by listing all files of the appropriate type `.txt` or `.zzz` extensions.
- check if the user-specified file name for the encrypting (or decrypting) exists before opening it for writing. If it already exists, ask the user if they are okay with it being wiped out. If they are not, ask them to pick a different file name to receive the encrypted (or decrypted) text.

Your program's interaction should looks something like: (Student input shown in **bold**.)

```
Welcome to the Encryption/Decryption Program

Would you like to (e)ncrypt a file, (d)ecrypt a file, or e(x)it (enter e, d, or x)? encrypt
Sorry, that's an invalid choice.  Please enter only e, d, or x: e

What positive integer key would you like to use for encryption? five
Sorry, that's an invalid choice.  Please enter a positive integer:  5

Enter the text-file name to encrypt: mesage.txt
Sorry the file 'mesage.txt' does NOT exist -- please try again!
Enter the text-file name to encrypt: messaeg.txt
Sorry the file 'messaeg.txt' does NOT exist -- please try again!
Enter the text-file name to encrypt: message.txt

The file 'message.txt' was successfully encrypted using a key of 5 to the file 'message.zzz'

Would you like to (e)ncrypt a file, (d)ecrypt a file, or e(x)it (enter e, d, or x)? d

What positive integer key would you like to use for decryption? 5

Enter the text-file name to decrypt: message.zzz

WARNING:  The file 'message.txt' already exists!
Is it okay to wipe it out (y/n)?  n
Enter the file name that should be used (.txt extension will automatically be added): decryptedMessage

The file 'message.zzz' was successfully decrypted using a key of 5 to the file 'decryptedMessage.txt'

Would you like (e)ncrypt a file, (d)ecrypt a file, or e(x)it (enter e, d, or x)? x
Bye!
```

When you write your program, be sure you:

- save your program in a file called `hw2.py`
- make your program robust by validating the correctness of user input: verify that user-entered file names exists (`import os.path`) before opening them, and check for correct menu-option, etc.
- think about the functional-decomposition (top-down) design **before you start to write code**! You will need to turn in a design document (see lab 1 for an example: diagram and sentence about each function).
- don't use global variables, except for global constants with good style (ALL_CAPS_AND_UNDERSCORES). Variables should be passed as parameters into functions and returned from functions
- use meaningful variable names with good style (i.e., useCamelCase)
- use comments (""" Multi-line Comment """) at the start of the program **and** immediately after each function `def`inition describing what they do (see lab1 `diceOutcomes.py` program)
- use a main function (see lab1 `diceOutcomes.py` program) located at the top of program **with a call to it at the bottom to start execution**

Submit a single zipped file called hw2.zip containing the following:

- **`hw2.py`** (your Python program)
- **design.doc** (or design.pdf, or design.txt, or design.rtf) a document describing the design of your program including a functional-decomposition diagram (**can be hand-drawn**) with text describing each function (see lab1 description)

The steps for the homework submission system are:

1. Design, write, debug, and test your program in the hw2 folder. When you are ready to submit your homework, zip the whole folder by right-clicking on it and selecting `Send to | Compressed (zipped) folder`. This will create a new file called `hw2.zip` which you will submit electronically. (This assumes Windows OS....)
2. Log on to the submission system at: **https://www.cs.uni.edu/~schafer/submit/which_course.cgi**
3. Select the course and section number of "CS 1520, Data Structures, Fienup". Click the "Continue".
4. Select the homework that you wish to submit: "HW 2: File Encryption/Decryption". Click the "Continue" button.
5. Specify how many extra files you want to submit. Just leave it at 0. Click the "Continue" button.
6. Upload your program by Browsing and selecting your `hw2.zip` file. Click the "Continue" button.
7. The next page reports on the status of the upload(s). You can always continue to upload a better version of the program until the deadline. The newer file will replace an older file of the same name.

(If you miss the deadline, you'll need to submit it as above, but select "Late Homeworks" in step 4 above.)