SY201 – Cyber Operations Fundamentals; Programming Assignment 05

Caesar Cipher

Name: _____ Alpha: _____

Begin this assignment by making sure your repo is up to date. Change to **~/repo201** and type:

git pull

All the files required for this assignment, including an electronic version of this handout, are available in:

~/repo201/programming/pa05

Read this entire handout at least once before you begin your research or start coding

1. **Background**:

"The Caesar cipher is one of the earliest known and simplest ciphers. It is a type of substitution cipher in which each letter in the plaintext is [rotated] a certain number of places down the alphabet. For example, with a [rotation] of 1, A would be replaced by B, B would become C, and so on. The method is named after Julius Caesar, who apparently used it to communicate with his generals."[1]

The rotation in a Caesar Cipher (which we'll call R) is an integer that represents how many positions to move from a plaintext character in the alphabet to get the corresponding ciphertext character. For example, if R = 4 (assume movement to the right), the string "NAVY" encrypts to "REZC":

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Note that we wrap around the alphabet if we get to the end. This is a perfect scenario for the mod (%) operator. It works like this:

a. Start with the letter 'N' and determine its position (P1) in the alphabet (13, if we start counting with 'A' at position 0).

b. Determine a new position (P2) as: P2 = (P1 + R) % 26. For our scenario that

[1] http://practicalcryptography.com/ciphers/caesar-cipher/

would be `P2 = (13 + 4) % 26 = 17`. If we look at position `17` we find `'R'`, so `'N'` encrypts to `'R'`.

c. In the string `"NAVY"` the `'Y'` demonstrates the power of the `mod` (`%`) function. The position of `'Y'` is `24` (starting from `0`). `P2 = (24 + 4) % 26 = 2`. If we look at position `2` we find `'C'`, so `'Y'` encrypts to `'C'`.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

2. **Your Program Should Do This:**

a. Present a menu and prompt the user to select one of three options by entering the appropriate integer:

```
*** Caesar Cipher ***

1. Encrypt a file
2. Decrypt a file
3. Quit

Option:
```

b. For option `1`:

i. Prompt the user for the name of a plaintext file containing a message.
ii. Prompt the user for the name of another file to be used to hold the encrypted version of the message.
iii. Prompt the user for a rotation value (`R`) in the range from `0` to `25`.
iv. Encrypt each letter in the plaintext by applying the rotation and save the resulting encrypted message to the file selected in part 2.b.ii, above.

c. For option `2`:

i. Prompt the user for the name of an encrypted file containing a coded message.
ii. Prompt the user for a rotation value (`R`) in the range from `0` to `25`.
iii. Apply the rotation value in reverse to decrypt each letter of the encrypted file and print the resulting plaintext message to the screen (*don't save it to a file*).

d. For option 3, exit your program normally.

3. **Design Requirements:**

a. Think carefully about how you want to approach this assignment before you start coding. It's rarely effective when you open up your editor and just start typing. Think first about the kinds of variables you will use, their names, and how the different sections of your code will interact. Take out a piece of paper and pencil and sketch out the major components of your design (collect input, validate input, perform calculations, present results, etc.). Draw simple flow diagrams or write code-like snippets (called *pseudocode*) that capture the major steps in your program. Your design sketch and / or pseudocode is a required deliverable for this assignment.

b. The encryption / decryption portion of your code will only need to handle uppercase letters and non-alphabetical special characters (punctuation, newline characters ('\n'), etc.). No lowercase alphabetical characters will be provided as input. Your program should encrypt / decrypt uppercase letters, but pass-through any non-alphabetical characters unchanged.

c. You must make use of functions in this assignment. At a minimum you must design and use the following three functions:

i. `def getInt(prompt, lower, upper):`

This function prompts the user for an integer in the range: [`lower,upper`], using `prompt` and returns a valid integer to the calling program. The function must notify the user if an invalid integer is entered (e.g. a string) or if the integer entered is not within [`lower,upper`]. For these cases, keep looping until the user provides compliant input.

ii. `def getFile(prompt, mode):`

This function prompts the user for a file to open using the provided `prompt` and `mode` ("`r`" or "`w`"). The function opens the file returns the file handle to the calling program. The function must notify the user if an error occurs when opening a file (e.g. trying to open a file that doesn't exist using mode "`r`"). For these cases, keep looping until the user provides compliant input.

iii. `def transcodeCharacter(ch, direction, R):`

This function transcodes a single character (either encrypts it or decrypts it). It returns the transcoded character to the calling program, based on the following three parameters:

1. `ch`: Character to be transcoded
2. `direction`: Either the string "`encrypt`" or the string "`decrypt`"
3. `R`: Rotation to be used in a Caesar Cipher

4. **Error Handling**

a. Each of the three required functions that you write must follow the error checking rules specified in part 3.c, above.

5. **Program Flow**

Here is a sample run of the program. This flows across two columns to keep everything on one page.

```
*** Caesar Cipher ***

1. Encrypt a file
2. Decrypt a file
3. Quit

Option: 1
Enter file to encrypt: first.txt
Enter output file: first05.enc
Enter R: 5

*** Caesar Cipher ***

1. Encrypt a file
2. Decrypt a file
3. Quit

Option: 2
Enter file to decrypt: first05.enc
Enter R: 5

GO NAVY!
```

```
*** Caesar Cipher ***

1. Encrypt a file
2. Decrypt a file
3. Quit

Option: 3
```

6. **Testing:**

   a. Thoroughly testing your program can be a difficult task. For complex programs there may be hundreds or thousands of possible cases to test. The directory for this project also includes several plaintext and encrypted files you can use to test your code. The encrypted file names are of the form: *nameDD.enc*, where `DD` represents the `R` value that was used to encode the file. For example: `first05.enc` is a file that was encoded using an `R` value of `5`.

   b. I recommend conducting runs on the test files using the executable version of the program; then run the test files through your version of the program and compare the results.

7. **Assumptions You May Make:**

   a. Plaintext files will only have uppercase letters and non-alphabetical special characters; no lowercase letters. Likewise for ciphertext files.

   b. While your user may end up entering an invalid number for rotation (`R`), and your `getInt()` function will have to handle that, you may assume that any encoded test file you encounter will have had a valid `R` [`0,25`] applied when it was transcoded from plaintext to ciphertext.

   c. When *encrypting* files, rotations move to the right (`+R`). For *decrypting* files, you need to move to the left (`-R`).

   d. If the user enters a filename with `mode "w"`, and the file already exists, the file will be overwritten upon being opened. No need to handle cases like: "*File already exists, are you sure you want to overwrite it?*"

8. **Hints:**

   a. Give your variables meaningful, descriptive names. In this case it's okay to use the variable named `R` for the rotation value.

b. When you *encode* a character you move "to the right" along the alphabet using the `mod` operator (`%`) to add `R` and wrap around to the beginning. When you *decode* a character you subtract `R` and move "to the left" along the alphabet. How can you wrap around from the beginning to the end if you subtract `R` and end up with a negative number?

c. The directory for this project also includes an executable version and several test files that you can run to get a sense for how your program should operate. You should also use these tools to test your own code. The program is called: `caesar`.

   Let's assume you're working in your `~/shares/sy201` directory. You can run `caesar` by first copying it to your working directory and then adjusting its permissions with the following commands:

   i. `cp ~/repo201/programming/pa05/caesar .`
      (*the trailing space and period in the command above are important*)
   ii. `chmod 755 caesar`

   You only have to perform steps (i) and (ii) once. After that, you can run the program any time you want by typing: `./caesar`

9. **Deliverables And Due Dates:**

a. Using paper, and pen or pencil, complete your pseudocode / flow diagram *before* you start coding. You may use more than one page for this if you need it. Be sure to indicate your name and alpha on this page.

b. Your completed source code (the one ending in `.py`).

c. Submit parts (a) and (b) by 2359 on Tuesday, 16 October, in accordance with your instructor's directions.