Programming Project 04

This assignment is worth 40 points (4.0% of the course grade) and must be **completed and turned in before 11:59 PM on Monday, October 2, 2023 (2 weeks because of the exam). After the due date, your score will be deducted by 25% for every 12 hours late or a fraction of it.**

You are not allowed to use advanced data structures such as lists, sets, dictionaries and classes. You are also not allowed to use the `chr()` function. Use of these prohibited concepts earns a zero points for the whole project.

1. Assignment Deliverable

The deliverable for this assignment is the following file:

`proj04.py` – the source code for your Python program

Be sure to use the specified file name and to submit it for grading via **Codio** before the project deadline.

2. Assignment Overview (learning objectives)

This assignment will give you more experience on the use of:

1. integers (int)
2. conditionals and iteration
3. string
4. function

The goal of this project is to implement a different cryptographic technique by combining two cryptographic techniques. We call it “dumbcrypt”. Dumbcrypt is a combination of Affine Cipher and Caesar Cipher. The user will enter a sentence, a string composed of letters, numbers, and punctuation. Notice that spaces are not allowed in the input string because spaces provide too many clues to someone trying to crack your coded message. Your job is to encrypt the sentence using a combination of Affine and Caesar Cipher. *The letters and numbers will be encrypted and decrypted using Affine Cipher; the punctuation using Caesar Cipher.*

3. Assignment Background

3.1 Affine Cipher - https://en.wikipedia.org/wiki/Affine_cipher

3.1.1 Affine Cipher Encryption

Affine Cipher uses an encryption function to calculate the integer that corresponds to the cipher text letter. The encryption function for a letter variable named *x* (the index of the letter in the alphabet) is:

$$\boxed{E(x) = (Ax+N) \bmod M}$$

Where A and N are keys of the cipher and M is the size of the alphabet (for example, M=26 for English letters; M = 36 for English letters plus digits). The decryption of Affine Cipher only works when A and M are co-primes of each other (check in the following page for the definition of Co-primes.

N will be an input to the program and will be fixed for the whole run of the program. N is sometimes referred to as the “rotation” but is effectively a key.

Co-primes are a pair of numbers whose greatest common divisor (GCD) is 1. Only one co-prime is needed (of possibly many). For this project, we choose the smallest greater than one so everyone is using the same one—making testing feasible.

Encryption Algorithm for Affine Cipher:

1. Calculate M, the number of characters in the alphabet (Hint: use len())
2. Calculate A using `get_smallest_co_prime(M)`
3. For the character to be encrypted **find** its index in the alphabet: *x* in the formula is the index.
4. Apply the formula to get the index of the cipher character: $E(x) = (Ax+N) \bmod M$
5. Using the index get the cipher character from the alphabet.
6. Return the character.

3.1.2 Affine Cipher Decryption

Decryption is a little more complicated than encryption. The decryption function is

$$\boxed{D(x) = A^{-1}(x-N) \bmod M}$$

Where A^{-1} is the modular multiplicative inverse (https://en.wikipedia.org/wiki/Modular_multiplicative_inverse). We provide the function `multiplicative_inverse(A,M)` that calculates the multiplicative inverse, A^{-1}, for you. The function checks all the numbers *x* from 1 to M and for every number *x* checks if *(A*x) mod M* is 1. Once you have the A^{-1}, you can use the decryption function to calculate the integer that corresponds to plaintext letter.

The decryption algorithm is the same as encryption with two modifications:

Decryption Algorithm for Affine Cipher:

1. Calculate M, the number of characters in the alphabet (Hint: use len())
2. Calculate A using `get_smallest_co_prime(M)`
3. Calculate A^{-1} using the function `multiplicative_inverse(A,M)`.
4. For the character to be encrypted **find** its index in the alphabet: *x* in the formula is the index.
5. Apply the formula to get the index of the cipher character: $D(x) = A^{-1}(x-N) \bmod M$
   Note the formula difference from encryption: the parentheses are different, and it has subtraction.
6. Using the index get the cipher character from the alphabet.
7. Return the character.

3.1.3 Example

Consider the alphabet being a single string consisting of the lower-case English letters as below (shown with each letter's associated index, e.g. the letter m's index is 12):

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
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

Based on the alphabet being used in this example, `M` is 26 since there are 26 characters in the alphabet; `A` is 3 since the smallest co-prime greater than one is 3.

To encrypt consider our plaintext 'green'. Using the affine key (A=3, N=8), we will encrypt "green" using the table above for the index values of each letter. The first step is to write the numeric values of each letter using the alphabet. Then, take each value of x, and solve the first part of the equation, (3x + 8). After finding the value of (3x + 8) for each character, take the remainder when dividing the result of (3x + 8) by 26. The final step in encrypting the message is to look up each index value in the table for the corresponding letters. In this example, the encrypted text would be "ahuuv". The table below shows the completed table for encrypting a message in the Affine cipher.

plaintext	g	r	e	e	n
x	6	17	4	4	13
(3x+8)	26	59	20	20	47
(3x+8) mod 26	0	7	20	20	21
ciphertext	a	h	u	u	v

To decrypt you want to reverse that process. That is, start with the ciphertext "a" and work backwards (using the affine mapping) to get "g" and repeat for the remaining characters.
Based on the alphabet being used in this example, `M` is 26; `A` is 3. Then, we need to find the multiplicative inverse (the function is provided to you) which is 9. In this example, the decrypted text would be "green". The table below shows the completed table for encrypting a message in the Affine cipher.

ciphertext	a	h	u	u	v
x	0	7	20	20	21
9*(x-8)	-72	-9	108	108	117
9*(x-8) mod 26	-20	-9	4	4	13
plaintext	g	r	e	e	n

Note that in this example, we use the string of the alphabet only. In this project, we will be using string with letters and digits. The string is provided in the starter code.

3.2 Caesar Cipher - https://en.wikipedia.org/wiki/Caesar_cipher

Caesar Cipher is a simplified form of Affine cipher which follows similar encryption and decryption algorithm. Caesar Cipher uses a single key to calculate the integer that corresponds to the ciphertext letter. The encryption function is:

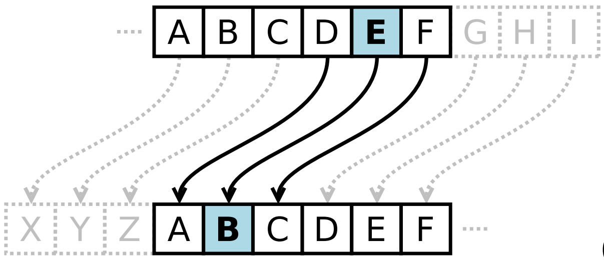
$$\boxed{E(x) = (x+N) \bmod M}$$

The decryption function is:

$$\boxed{D(x) = (x-N) \bmod M}$$

The algorithms for Caesar Cipher are similar to the Affine Cipher algorithms. Consider the following example. Then, if we assume N=23, the new Cipher alphabet is as follows:

x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
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



(source: https://en.wikipedia.org/wiki/Caesar_cipher)

Using the example above, the word “green” is translated as follows:

- the ‘g’ is found at index 6 in the alphabet. The letter in the Cipher alphabet is ‘d’.
- the ‘r’ is found at index 17, the rotated letter is ‘o’
- the ‘e’ is found at index 4, the rotated letter is ‘b’
- the ‘e’ is found at index 4, the rotated letter is ‘b’
- the ‘n’ is found at index 13, the rotated letter is ‘k’

To decrypt you want to reverse that process. That is, start with the ciphertext “d” and work backwards (using the Caesar mapping) to get “g” and repeat for the remaining characters.

4. Project Description

Your program must meet the following specifications:

1. You have to use the eight functions in the provided `proj04.py` skeleton. You have to implement and use the following:
    a. `check_co_prime(num, M):` Takes two numbers as parameters. Returns `True` if num and M are co-primes, otherwise returns `False`. (Hint: used GCD). Check the "assignment background" section for the definition of a coprime.
    b. `get_smallest_co_prime(M):` Takes one number M as a parameter: Returns the smallest coprime of M that is greater than 1. For example, if M = 26, the smallest co-prime greater than one is 3. If M = 210, the smallest co-prime greater than one is 11.
    (Hint: use `check_co_prime`)
    c. `caesar_cipher_encryption(ch,N,alphabet):` Takes three inputs: a character to encrypt, the rotation N, and the alphabet. Returns the cipher text character using the Caesar Cipher.
    d. `caesar_cipher_decryption(ch,N, alphabet):` Takes three inputs: a character to encrypt, the rotation N, and the alphabet. Returns the plain text character using the Caesar Cipher.
    e. `affine_cipher_encryption(ch,N, alphabet):` Takes three inputs: a character to encrypt, the rotation N, and the alphabet. Returns the cipher text character using the Affine Cipher.
    f. `affine_cipher_decryption(ch,N, alphabet):` Takes three inputs: a character to encrypt, the rotation N, and the alphabet. Returns the plain text character using the Affine Cipher.
    g. `main():` Takes no input. Returns nothing. First prompts for rotation, N. Then prompts for a command (d,e,q), prompts for a string, decrypts or encrypts the string depending on the command using the encryption and decryption functions described above. Remember to use a different encryption/decryption algorithm for punctuation. We will be using the Affine Cipher for letters and numbers, and the Caeser Cipher for punctuation. We provide two constant strings in the starter code to use.

5. Assignment Notes and Other Requirements

1. To clarify the project specifications, sample output is appended to the end of this document.
2. Convert all letters to lower case before encryption or decryption.
3. Items 1-8 of the Coding Standard will be enforced for this project.
4. We provide a `Project04.zip` file for you to start with. You need to download the file, unzip the file and open it in PyCharm. The zip file contains the function tests as well as the text files for the input/output tests.
5. You may import the GCD function from a module such as math.
6. You are not allowed to use advanced data structures such as lists, sets, dictionaries and classes.
7. You are not allowed to use the `chr()` function.
8. There are two alphabets provided: `ALPHA_NUM` (numbers + letters) and `PUNCTUATION` (string.punctuation).
9. The *mod* operator in the encryption formulas is the % operator in Python.
10. The string method `isalnum()` returns `True` if the string is composed of letters and digits.
11. The N in the formulas is often referred to as the "rotation" and is the first value prompted for. It effectively is the key for encryption and decryption. We will be using the same N for both encryptions.

12. Error checking (Hint: do error checking after everything else is working.)
    a. Re-prompt if the rotation is not an int.
    b. Print an error message and ignore any commands that are not ‘e’, ‘d’ or ‘q’.
    c. Print an error message if any of the text is not a letter, digit or punctuation. Note that a space will generate an error (spaces provide too many clues for cracking codes).
13. A strings.txt file of strings used is provided to help with testing. The same strings are provided in the starter code.
14. I found `else` with the `for` statement to be useful to not print plain and cipher texts if there was an error with a character.

6. Grading Rubric

```
5 pts Coding Standard 1-8
     (Descriptive comments, mnemonic identifiers, format, etc...)

2 pts check_co_prime

2 pts get_smallest_co_prime

2 pts caesar_cipher_encryption

2 pts caesar_cipher_decryption

2 pts affine_cipher_encryption

2 pts affine_cipher_decryption

12 pts Blind Functions Test (partial credit for each function

3 pts Test 1

3 pts Test 2

5 pts Test 3 Blind Test
```

Note:

- hard coding an answer earns zero points for the whole project.
- -10 points for not using main() in a meaningful operations.
- Use of any advanced data structures (such as lists, sets, dictionaries and classes) or the use of chr() function earns zero points for the whole project.

7. Test Cases

7.1 Function Unit Tests

Check the test files in the Project04 starter package. when you run one test file as you normally run a python file, if an assertion error is generated, it means you failed the test. The test will show you the instructor’s answer and your answer.

7.2 Input/Output Tests

Test 1

```
**************************************************
*   Welcome to the world of 'dumbcrypt,' where cryptography meets comedy!
    We're combining Affine Cipher with Caesar Cipher to create a code
    so 'dumb,' it's brilliant.
    Remember, in 'dumbcrypt,' spaces are as rare as a unicorn wearing a top hat!
    Let's dive into this cryptographic comedy adventure! *
**************************************************

Input a rotation (int): 4


Input a command (e)ncrypt, (d)ecrypt, (q)uit: e

Input a string to encrypt: Hello,World!

Plain text: Hello,World!

Cipher text: dyxxc:gcrxt%


Input a command (e)ncrypt, (d)ecrypt, (q)uit: d

Input a string to decrypt: dyxxc:gcrxt%

Cipher text: dyxxc:gcrxt%

Plain text: hello,world!


Input a command (e)ncrypt, (d)ecrypt, (q)uit: e

Input a string to encrypt: Fahrenheit-451?

Plain text: Fahrenheit-451?

Cipher text: 3edry7dyi1;kp5]


Input a command (e)ncrypt, (d)ecrypt, (q)uit: d

Input a string to decrypt: 3edry7dyi1;kp5]

Cipher text: 3edry7dyi1;kp5]

Plain text: fahrenheit-451?


Input a command (e)ncrypt, (d)ecrypt, (q)uit: q
```

Test 2

```
**************************************************
*   Welcome to the world of 'dumbcrypt,' where cryptography meets comedy!
    We're combining Affine Cipher with Caesar Cipher to create a code
    so 'dumb,' it's brilliant.
    Remember, in 'dumbcrypt,' spaces are as rare as a unicorn wearing a top hat!
    Let's dive into this cryptographic comedy adventure! *
**************************************************

Input a rotation (int): x

Error; rotation must be an integer.
Input a rotation (int): 1.2

Error; rotation must be an integer.
Input a rotation (int): 8


Input a command (e)ncrypt, (d)ecrypt, (q)uit: d

Input a string to decrypt: cg)cv22b+

Cipher text: cg)cv22b+

Plain text: go!green#


Input a command (e)ncrypt, (d)ecrypt, (q)uit: e

Input a string to encrypt: hello world

Error with character:
Cannot encrypt this string.


Input a command (e)ncrypt, (d)ecrypt, (q)uit: q
```

Test 3 Blind Test