

# CECS 229 Programming Assignment #3

## Due Date:

Sunday, 3/10 @ 11:59 PM

## Submission Instructions:

Complete the programming problems in the file named `pa3.py`. You may test your implementation on your Repl.it workspace by running `main.py`. When you are satisfied with your implementation,

1. Submit your Repl.it workspace
2. Download the file `pa3.py` and submit it to the appropriate CodePost auto-grader folder.

## Objectives:

1. Encrypt and decrypt text using an affine transformation.
2. Encrypt and decrypt text using the RSA cryptosystem.

## NOTES:

1. Unless otherwise stated in the FIXME comment, you may not change the outline of the algorithm provided by introducing new loops or conditionals, or by calling any built-in functions that perform the entire algorithm or replaces a part of the algorithm.
2. You may use the utility functions found in `util.py`.
3. You may use any functions you implemented in previous programming assignments for this course. If you choose to use them, please make sure to copy and paste their implementation into `pa3.py` file, so that they are uploaded to CodePost when you submit your work.

---

## Problem 1:

Complete the function `affine_encrypt(text, a, b)` that returns the cipher text encrypted using key `(a, b)`. You must verify that the  $\gcd(a, 26) = 1$  before making the encryption. If  $\gcd(a, 26) \neq 1$ , the function must raise a `ValueError` exception with message "The given key is invalid."

```
In [ ]: def affine_encrypt(text, a, b):  
        """  
        encrypts the plaintext 'text', using an affine transformation key (a, b)  
        :param: text - str type; plaintext as a string of letters  
        :param: a - int type; integer satisfying gcd(a, 26) = 1
```

```

:param: b - int type; shift value
:raise: ValueError if gcd(a, 26) is not 1.
:return: str type; the encrypted message as string of uppercase letters
"""

# FIXME: raise an error if the gcd(a, 26) is not 1

cipher = ""
for letter in text:
    if letter.isalpha():
        # FIXME: Use util.py to initialize 'num' to be
        # the integer corresponding to the current letter
        num = None

        # FIXME: Encrypt the current 'num' using the
        # affine transformation with key (a, b).
        # Store the result in cipher_digits.
        cipher_digits = 'None'

        if len(cipher_digits) == 1:
            # FIXME: If the cipherdigit is 0 - 9,
            # prepend the string with a 0
            # to make it a two-digit number
            cipher_digits = None

        # FIXME: Use util.py to append to the cipher the ENCRYPTED letter
        # corresponding to the current cipher digits
        cipher += 'None'

return cipher

```

## Problem 2:

Complete the function `affine_decrypt(ciphertext, a, b)` that decrypts the text given in `ciphertext` which was encrypted using key `(a, b)`. You must verify that the  $\gcd(a, 26) = 1$ . If  $\gcd(a, 26) \neq 1$ , the function must raise a `ValueError` exception with message "The given key is invalid."

```

In [ ]: def affine_decrypt(ciphertext, a, b):
        """
        decrypts the given cipher, assuming it was encrypted using an affine transformatic
        :param: ciphertext - str type; a string of digits
        :param: a - int type; integer satisfying gcd(a, 26) = 1.
        :param: b - int type; shift value
        :return: str type; the decrypted message as a string of uppercase letters
        """

        a_inv = 0 # FIXME: complete this line so that a_inv holds the inverse of a under m

        text = ""
        for letter in ciphertext:
            if letter.isalpha():
                letter = letter.upper()

            # FIXME: Use util.py to find the integer `num` that corresponds
            # to the given letter

```

```

num = None

# FIXME: Decrypt the integer that corresponds to the current
# encrypted letter using the decryption function for an affine
# transformation with key (a, b) so that letter_digits holds
# the decrypted number as a string of two digits
letter_digits = 'None'

if len(letter_digits) == 1:
    # FIXME: If the letter number is between 0 - 9, inclusive,
    # prepend the string with a 0
    letter_digits = None

# FIXME: Use util.py to append to the text the decrypted
# letter corresponding to the current letter digits
text += 'None'

return text

```

### Problem 3:

Complete the function `encryptRSA(text, n, e)` which uses RSA to encrypt the string `text` using key `(n, e)`.

EXAMPLE

```
>> encryptRSA("REPEAT", 2537, 13)
```

```
'194319342299'
```

```

In [ ]: def encryptRSA(plaintext, n, e):
        """
        encrypts plaintext using RSA and the key (n, e)
        :param: text - str type; plaintext as a string of letters
        :param: n - int type; positive integer that is the modulo in the RSA key
        :param: e - int type; positive integer that is the exponent in the RSA key
        :return: str type; the encrypted message as a string of digits
        """

        text = plaintext.replace(' ', '') # removing whitespace

        # FIXME: Use util.py to initialize 'digits' as a string of
        # the two-digit integers that correspond to the letters of 'text'
        digits = 'None'

        # FIXME: Use util.py to initialize 'l' with the length of each RSA block
        l = 0

        # FIXME: Use a loop to pad 'digits' with enough 23's (i.e. X's)
        # so that it can be broken up into blocks of length l

        # creating a list of RSA blocks
        blocks = [digits[i:i + l] for i in range(0, len(digits), l)]

        cipher = ""
        for b in blocks:
            # FIXME: Initialize 'encrypted_block' so that it contains

```

```

# the encryption of block 'b' as a string
encrypted_block = 'None'

if len(encrypted_block) < l:
    # FIXME: If the encrypted block contains less digits
    # than the block size l, prepend the block with enough
    # 0's so that the numeric value of the block
    # remains the same, but the new block size is l,
    # e.g. if l = 4 and encrypted block is '451' then prepend
    # one 0 to obtain '0451'
    encrypted_block = None

# FIXME: Append the encrypted block to the cipher
cipher += 'None'
return cipher

```

### Problem 4:

Complete the implementation of the function `decryptRSA(cipher, p, q, e)` which decrypts `cipher`, assuming it was encrypted using RSA and key  $(n = p \cdot q, e)$ .

EXAMPLE:

```
>> decryptRSA('03412005', 43, 59, 23)
```

STOP

```

In [ ]: def decryptRSA(cipher, p, q, e):
        """
        decrypts the cipher, which was encrypted using RSA and the key (p * q, e)
        :param: cipher - ciphertext as a string of digits
        :param: p, q - prime numbers used as part of the key n = p * q to encrypt the cipher
        :param: e - integer satisfying gcd((p-1)*(q-1), e) = 1
        :return: str type; the decrypted message as a string of letters
        """

        n = p * q
        ciphertext = cipher.replace(' ', '')

        # FIXME: Use util.py to initialize `l` with the size of
        # each RSA block
        l = 0

        # FIXME: Use a Python List comprehension to break the ciphertext
        # into blocks of equal length 'l'. Initialize 'blocks' so that it
        # contains these blocks as elements
        blocks = []

        text = "" # initializing the variable that will hold the decrypted text

        # FIXME: Compute the inverse of e
        e_inv = None

        for b in blocks:
            # FIXME: Use the RSA decryption function to decrypt
            # the current block

```

```
decrypted_block = 'None'

if len(decrypted_block) < l:
    # FIXME: If the decrypted block contains less digits
    # than the block size l, prepend the block with
    # enough 0's so that the numeric value of the block
    # remains the same, but the new block size is l,
    # e.g. if l = 4 and decrypted block is '19' then prepend
    # two 0's to obtain '0019'
    decrypted_block = None

    # FIXME: Use util.py to append to text the decrypted block
    # transformed into letters
    text += 'None'

return text
```