Student Name: Jonathan Riordan
Student ID: C13432152
Lab 1

Part 1 - Ceasar Cipher

Plaintext is entered in by the user, in my case, I entered in “hello”
The key is 1.

The Caesar cipher is a shifting algorithm. Each character in the plaintext is shifted by the key to encrypt. To Decrypt the cipher text. Each cipher text is shifted minus the the key.

The output of of running my code is.
Enter your message: hello
Encrypted message is: ifmmp
Decrypted message is: hello

Code

```
'''
   Advanced Security
   Lab 1
   Jonathan Riordan
   C13432152

   Part 1
'''

def getMessage():
        input_variable = raw_input("Enter your message: ")
        return input_variable

def caesar(s,k,decrypt=False):
        if decrypt: k=26-k
        r=""
        for i in s:
                if(ord(i) >= 65 and ord(i) <= 90):
                        r += chr((ord(i) - 65 + k) % 26 + 65)

                elif (ord(i) >= 97 and ord(i) <= 122):
                        r += chr((ord(i) - 97 + k) % 26 + 97)

                else:
                        r += i

        return r

def encrypt(p,k):
        return caesar(p,k)

def decrypt(c,k):
        return caesar(c,k,True)

message = getMessage()
key = 1
encrypted_message = encrypt(message, key)
print('Encrypted message is: ' + encrypted_message)
print('Decrypted message is: ' + decrypt(encrypted_message, key))
```

Part 2 - Rail fence

The plaintext is “hello”
The key is 2.

The following functions were included to make the rail fence cipher encrypt and decrypt.
To make the following program run, a message and a key is passed into the encryptR function.
To decrypt, the cipher text and the key is passed into the decryptR function.
The rail fence is a transposition cipher.

The output of the code is:
Encrypted message is: hloel
Decrypted message is: hello

Code

```
'''
Advanced Security
Lab 1
Jonathan Riordan
C13432152
Part 2
'''

def fence(p, k):
    fence = [[None] * len(p) for n in range(k)]
    rails = range(k - 1) + range(k - 1, 0, -1)
    for n, x in enumerate(p):
        fence[rails[n % len(rails)]][n] = x
    return [c for rail in fence for c in rail if c is not None]

def encryptR(p, k):
    return ''.join(fence(p, k))

def decryptR(c, k):
    rng = range(len(c))
    pos = fence(rng, k)
    return ''.join(c[pos.index(k)] for k in rng)

message = "hello"
key = 2

encrypted_message = encryptR(message, key)
print('Encrypted message is: ' + encrypted_message)
print('Decrypted message is: ' + decryptR(encrypted_message, key))
```