Homework Number: hw01
Name: Shu Hwai Teoh
ECN Login: teoh0
Due Date: Thursday 1/23/2020 at 4:29PM

1. Recovered plaintext quote:
   It is my belief that nearly any invented quotation, played with confidence, stands a good chance to deceive.
   - Mark Twain
2. Encryption key
   I. Binary: 0110 0010 0111 0010
   II. Decimal: 25202
3. Explanation
   To recover both the original quote and the encryption key, I did the following steps:
   I. Reduce the PassPhrase ("Hopes and dreams of a million years") to a bit array of size BLOCKSIZE (16) as the first “previous_decrypted_block” for decryption.
   II. Create a bitVector from the ciphertext hex string in ciphertextFile (encrypted.txt).
   III. Carry out differential XORing of bit blocks and decryption for all 2^{16} possible keys to find the key.
4. Code (on the second page)

```
from BitVector import *


PassPhrase = "Hopes and dreams of a million years"
BLOCKSIZE = 16
numbytes = BLOCKSIZE // 8

def cryptBreak(ciphertextFile,key_bv):
    # Reduce the PassPhrase to a bit array of size BLOCKSIZE:
    bv_iv = BitVector(bitlist=[0] * BLOCKSIZE)
    for i in range(0, len(PassPhrase) // numbytes):  # (G)
        textstr = PassPhrase[i * numbytes:(i + 1) * numbytes]  # (H)
        bv_iv ^= BitVector(textstring=textstr)

    # Create a bitvector from the ciphertext hex string:
    FILEIN = open(ciphertextFile)
    encrypted_bv = BitVector(hexstring=FILEIN.read())

    # Try all 2**16 possible keys to find the key
    for i in range(2 ** 16):
        previous_decrypted_block = bv_iv
        key_bv = BitVector(intVal=i, size=16)
        # Create a bitvector for storing the decrypted plaintext bit array:
        msg_decrypted_bv = BitVector(size=0)
        # Carry out differential XORing of bit blocks and decryption:
        for j in range(0, len(encrypted_bv) // BLOCKSIZE):
            bv = encrypted_bv[j * BLOCKSIZE:(j + 1) * BLOCKSIZE]
            temp = bv.deep_copy()
            bv ^= previous_decrypted_block
            previous_decrypted_block = temp
            bv ^= key_bv
            msg_decrypted_bv += bv
        # Extract plaintext from the decrypted bitvector:
        decryptedMessage = msg_decrypted_bv.get_text_from_bitvector()
        if 'Mark Twain' in decryptedMessage:
            #print("binary:", key_bv)
            #print("decimal:", i)
            #print(decryptedMessage)
            break
    return decryptedMessage

if __name__ == '__main__':
    someRandomInteger = 9999  # Arbitrary integer for creating a BitVector
    key_bv = BitVector(intVal=someRandomInteger, size=16)
    decryptedMessage = cryptBreak('encrypted.txt', key_bv)
    if 'Mark Twain' in decryptedMessage:
        print('Encryption Broken!')
    else:
        print('Not decrypted yet')
```