Spec Sheet

DES takes in a 64-bit binary message

DES takes in an 8-byte key (expressed in bytes)

I need to convert 8-byte initialization vector from HEX to a binary so I can XOR with the plaintext binary input.

When possible, all binaries should be expressed as lists.

Part 1)

1. ~~Create XOR method for CBC block method DES~~
2. ~~Break message into 8 byte chunks (64 bits)~~
3. Execute block DES on message
4. Output results for encryption
5. Do similar functionality for decryption

Estimated: 7 hours

Part 2)

Seems pretty straightforward. Just gotta write a few methods to extract parity bits and calculate new keys.

Estimated: 10 hours

Part 3) Fill in the blank, easy peasy. Estimated, 1 hour

THIS IS SUPER HELPFUL

nlee-osx:task1 nlee$ python

Python 2.7.10 (default, Jul 14 2015, 19:46:27)

[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.39)] on darwin

Type "help", "copyright", "credits" or "license" for more information.

>>> import binascii

>>> binascii.hexlify('12345678')

'3132333435363738'

>>> bin(int('3132333435363738',16))[2:]

'11000100110010001100110011010000110101001101100011011100111000'

>>>hex(int('11000100110010001100110011010000110101001101100011011100111000',2))[2:]

'3132333435363738'

>>> binascii.unhexlify('3132333435363738')

'12345678'

PART 2

1. Take in HEX and convert to binary (64 bit zfill)
2. Break it into 8 bit chunks
3. Extract parity bits (lowest bit on each array)
4. Compile it into a binary
5. Add 1 to the binary
6. Split it back into 7 bit chunks after increment
7. Add the parity bits back via ODD Parity formulae

268435456 (number of keys tried)

Consumed CPU time=20685.348457

Average number per minute = 778624.899338

268435456

Consumed CPU time=21421.819959

Average number per minute = 751856.163056

Fill in the Blank

1. 20685

2. 5552587407360

3. 28830682151975371230166645779358209859450306560

4. 8922663948960175746442995313200016967746949318084027868346050848080527360

5. Yes

6. 0.00000000002771394225220547014032490551471710205078125

7. Yes

8. 0.0000000000000000000000000000000000000029