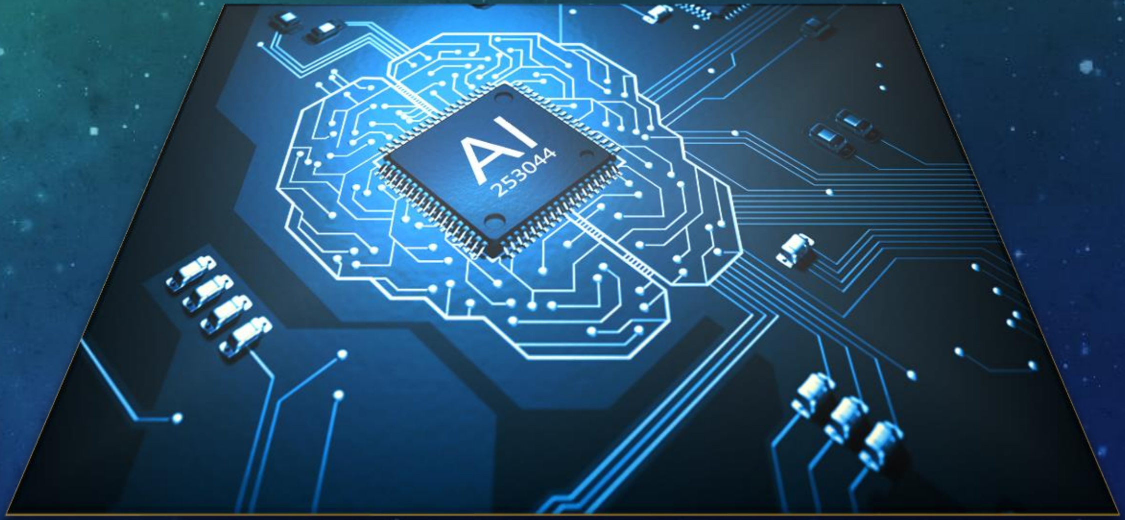
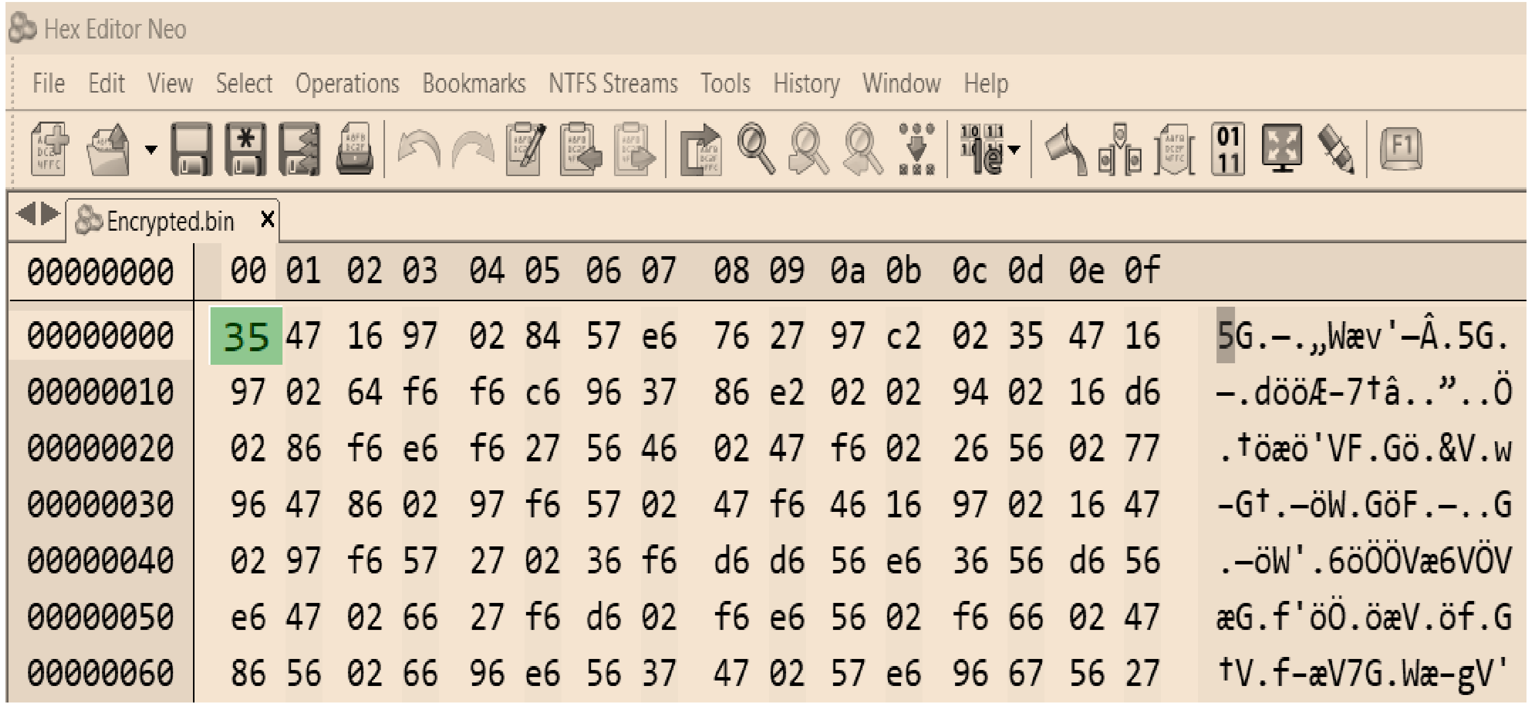
EXERCISE AI



Specification

- Purpose: Decrypt a byte rotated file

Byte	Shift >>	Char
0011 0101	0	5
1001 1010	1	š
0100 1101	2	M
1010 0110	3	¦
0101 0011	4	S
1010 1001	5	©
1101 0100	6	Ô
0110 1010	7	j


Hex Editor Neo
File Edit View Select Operations Bookmarks NTFS Streams Tools History Window Help
Encrypted.bin

```
00000000  00 01 02 03 04 05 06 07  08 09 0a 0b 0c 0d 0e 0f
00000000  35 47 16 97 02 84 57 e6  76 27 97 c2 02 35 47 16   5G.–.„Wæv'–Â.5G.
00000010  97 02 64 f6 f6 c6 96 37  86 e2 02 02 94 02 16 d6   –.dööÆ–7†â..”..Ö
00000020  02 86 f6 e6 f6 27 56 46  02 47 f6 02 26 56 02 77   .†öæö'VF.Gö.&V.w
00000030  96 47 86 02 97 f6 57 02  47 f6 46 16 97 02 16 47   –G†.–öW.GöF.–..G
00000040  02 97 f6 57 27 02 36 f6  d6 d6 56 e6 36 56 d6 56   .–öW'.6öÖÖVæ6VÖV
00000050  e6 47 02 66 27 f6 d6 02  f6 e6 56 02 f6 66 02 47   æG.f'öÖ.öæV.öf.G
00000060  86 56 02 66 96 e6 56 37  47 02 57 e6 96 67 56 27   †V.f–æV7G.Wæ–gV'
```

Bit Rotating 240: 11110000

0 <<<: 11110000	0 >>> 11110000
1 <<<: 11100001	1 >>> 01111000
2 <<<: 11000011	2 >>> 00111100
3 <<<: 10000111	3 >>> 00011110
4 <<<: 00001111	4 >>> 00001111
5 <<<: 00011110	5 >>> 10000111
6 <<<: 00111100	6 >>> 11000011
7 <<<: 01111000	7 >>> 11100001
8 <<<: 11110000	8 >>> 11110000

Encryption Algorithm

```
Algorithm Encrypt(byte,amount)
        mask = 2^amount - 1
        lowbyte = byte & mask
        highshift = byte >> amount
        lowshift = lowbyte << (8-amount)
        wholebyte = lowshift + highshift
        return wholebyte
Usage:
        byte = ‘S’ <83 ascii>
        amount = 3
        Encrypt(byte,amount)
```

```
Sample Data (0b01010011 (83) , 3) 0b01010011
        mask = 2^3 – 1 =               0b00000111
        lowbyte = 83 & 7 =             0b00000011
        highshift = 83 >> 3 =          0b00001010
        lowshift = lowbyte << 5 =      0b01100000
        wholebyte =                    0b01101010
        return wholebyte
```

Decryption Algorithm

```
Algorithm Decrypt(byte,amount)
    // Reverse the Encrypt process
Usage:
    byte = 'S' <83 ascii>
    amount = 3
    Decrypt(byte,amount)
```

During the decrypt process, the amount-to-rotate isn’t known where the amount to rotate ranges from 1 to 7.

Easy Hack Solution

- Since the rotate amount is unknown during the Decrypt process, iterate through the file with rotations 1 thru 7 printing each decrypted character.
- Then inspect the output and determine which rotation produces readable results.
- This hack requires human intervention to determine which rotation produced the readable results.
- Human Intervention in interpreting the results is only a last resort when there is no computer solution.

Computing Rotation Algorithm

This is a potential solution though not the ONLY solution. Use it in-case you don’t have an idea where to start.

A. See the Frequencies.txt in the zip file to determine the most frequent character

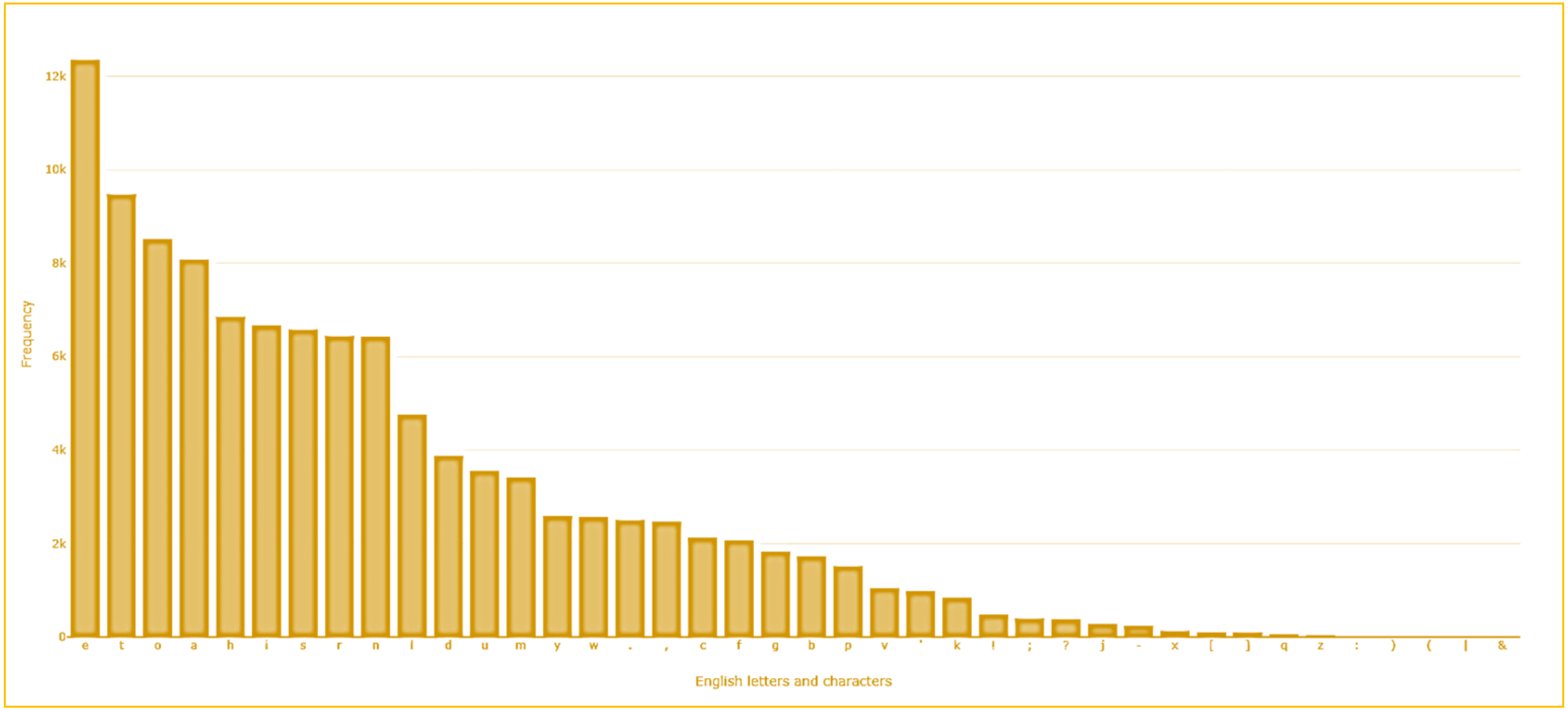
B. Do a rotation, and decrypt the file

C. Determine the most frequent decrypted character in the file

D. If the most frequent decrypted character is the same as the one in Frequencies.txt the rotation is correct

E. If these characters are different, repeat from step B

Example Character Frequency Barchart



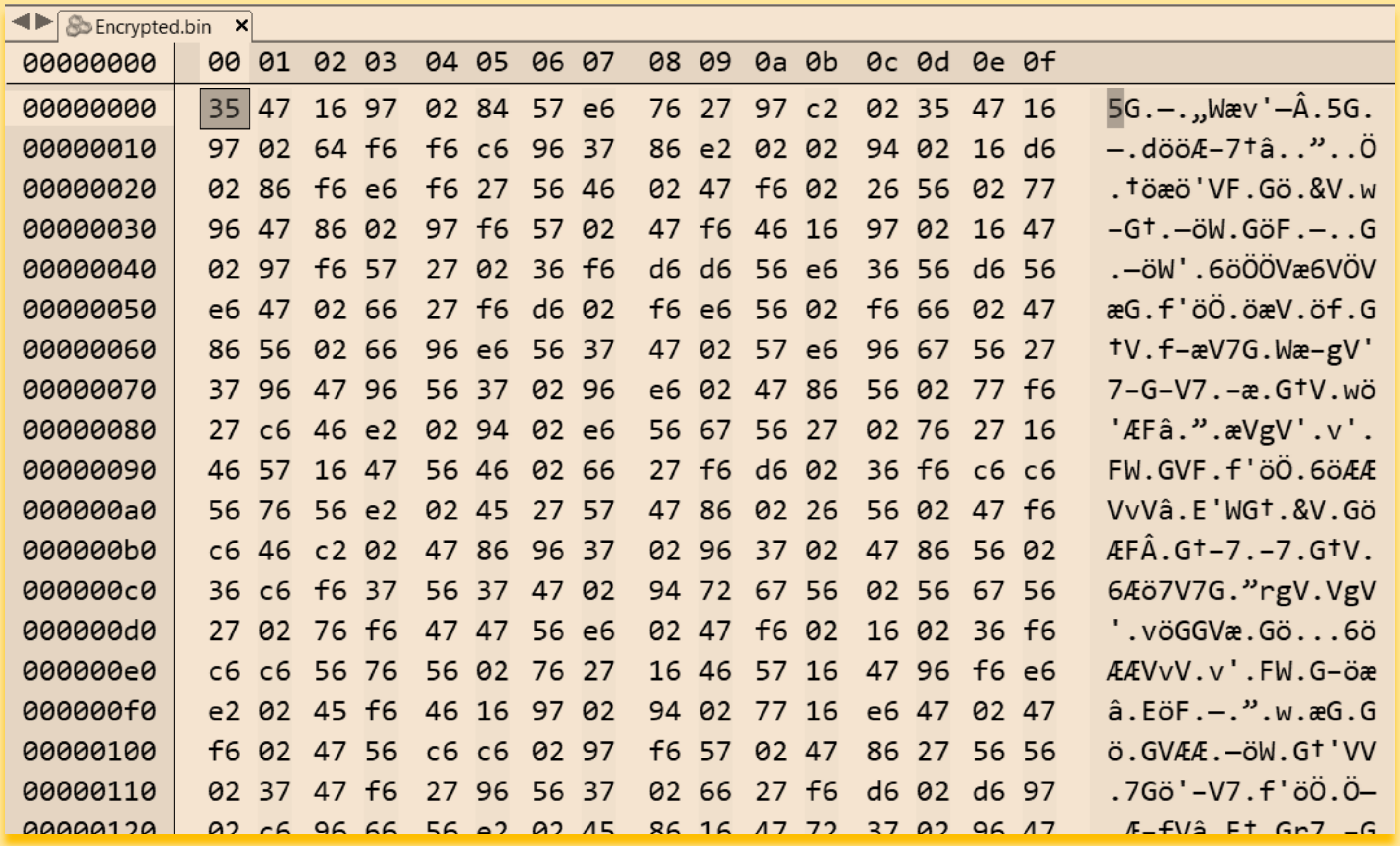
Failure Cases

- In the cases where the Computing Rotation Algorithm doesn’t produce the correct results, this means the Frequencies.txt character and the frequent decrypt character don’t match.
- At this point, you will need to refine the initial Computing Rotation Algorithm.
- The refinements are up to you to develop.

Data Files

The file to decrypt is:

Encrypted.bin



```
00000000   00 01 02 03  04 05 06 07  08 09 0a 0b  0c 0d 0e 0f
00000000   35 47 16 97  02 84 57 e6  76 27 97 c2  02 35 47 16   5G.–.„Wæv'–Â.5G.
00000010   97 02 64 f6  f6 c6 96 37  86 e2 02 02  94 02 16 d6   –.dööÆ–7†â..”..Ö
00000020   02 86 f6 e6  f6 27 56 46  02 47 f6 02  26 56 02 77   .†öæö'VF.Gö.&V.w
00000030   96 47 86 02  97 f6 57 02  47 f6 46 16  97 02 16 47   –G†.–öW.GöF.–..G
00000040   02 97 f6 57  27 02 36 f6  d6 d6 56 e6  36 56 d6 56   .–öW'.6öÖÖVæ6VÖV
00000050   e6 47 02 66  27 f6 d6 02  f6 e6 56 02  f6 66 02 47   æG.f'öÖ.öæV.öf.G
00000060   86 56 02 66  96 e6 56 37  47 02 57 e6  96 67 56 27   †V.f–æV7G.Wæ–gV'
00000070   37 96 47 96  56 37 02 96  e6 02 47 86  56 02 77 f6   7–G–V7.–æ.G†V.wö
00000080   27 c6 46 e2  02 94 02 e6  56 67 56 27  02 76 27 16   'ÆFâ.”.æVgV'.v'.
00000090   46 57 16 47  56 46 02 66  27 f6 d6 02  36 f6 c6 c6   FW.GVF.f'öÖ.6öÆÆ
000000a0   56 76 56 e2  02 45 27 57  47 86 02 26  56 02 47 f6   VvVâ.E'WG†.&V.Gö
000000b0   c6 46 c2 02  47 86 96 37  02 96 37 02  47 86 56 02   ÆFÂ.G†–7.–7.G†V.
000000c0   36 c6 f6 37  56 37 47 02  94 72 67 56  02 56 67 56   6Æö7V7G.”rgV.VgV
000000d0   27 02 76 f6  47 47 56 e6  02 47 f6 02  16 02 36 f6   '.vöGGVæ.Gö...6ö
000000e0   c6 c6 56 76  56 02 76 27  16 46 57 16  47 96 f6 e6   ÆÆVvV.v'.FW.G–öæ
000000f0   e2 02 45 f6  46 16 97 02  94 02 77 16  e6 47 02 47   â.EöF.–.”.w.æG.G
00000100   f6 02 47 56  c6 c6 02 97  f6 57 02 47  86 27 56 56   ö.GVÆÆ.–öW.G†'VV
00000110   02 37 47 f6  27 96 56 37  02 66 27 f6  d6 02 d6 97   .7Gö'–V7.f'öÖ.Ö–
00000120   02 c6 96 66  56 e2 02 45  86 16 47 72  37 02 96 47   .Æ–fVâ.E†.Gr7.–G
```

Specification

- Write classes to support the Decrypt and the Rotation algorithms.
- Use Bitsets, Bytes, where bit manipulations are needed.
- Use Regular Expressions to parse and search strings.
- Write any supporting classes necessary to support the Decrypt and Rotation Algorithms.
- For example, the Decrypt Algorithm needs support for reading an encrypted file.
- The Decrypt class is only responsible for decrypting a character, NOT reading a file AND decrypting each character.

0011 0101	>> 0	5
1001 1010	>> 1	š
0100 1101	>> 2	M
1010 0110	>> 3	¦
0101 0011	>> 4	S
1010 1001	>> 5	©
1101 0100	>> 6	Ô
0110 1010	>> 7	j