CET2894 – Cryptography Assignment

Student: ________________________ Date: ____________

Directions: Copy the question, then provide your answer in RED along with the method and tool/command used. Include a screenshot for full credit.

Challenge 1

Question:

```
3c 5c 64 5a 6d 40 72 35 34 59 3b 2d 2c 56 6f 40 54 51 24 41 40 39 35 70 35 40
73 4c 4c 55 39 52 2f 72 6e 40 50 5e 6b 6f 40 37 4e 65 35 38 50 21 42 32 41 34
28 51 52 41 32 37 68 3b 40 52 6a 46 32 3d 27 25 6b 33 40 35 39 66 54 41 51 4d
3f 41 40 73 4c 4c 39
```

Answer:

[Fill in after running the method below and taking a screenshot]

How it was solved / Type of encoding:

Likely ASCII-based shift/XOR (e.g., ROT47/byte-wise Caesar). Try: 1) remove spaces, 2) hex→bytes, 3) attempt ROT47 / ASCII Caesar over 94 printable characters, or single-byte XOR. Hint: CyberChef recipe: From Hex → ROT47 or XOR Brute Force.

Commands / Tools used:

```
# Python brute idea:

python3 - << 'PY'
h='3c5c645a6d40723534593b2d2c566f405451244140393570354073 4c4c5539522f726e40505
e6b6f40374e65353850214232413428515241323768 3b40526a46323d27256b334035396654415
14d3f4140734c4c39'
c=bytes.fromhex(h)
for k in range(256):
    try:
        s=bytes(b^k for b in c).decode('ascii')
        if all(32<=ord(ch)<=126 for ch in s):
            print(k, s)
    except: pass
PY
```

Challenge 2

Question:

```
47%2065%2065%202c%2020%2042%2072%2061%2069%206e%202c%2020%2077%2068%2061%2074%
2020%2064%206f%2020%2079%206f%2075%2020%2077%2061%206e%2074%2020%2074%206f%202
0%2064%206f%2020%2074%206f%206e%2069%2067%2068%2074%203f
```

Answer:

Gee, Brain, what do you want to do tonight?

How it was solved / Type of encoding:

URL encoding → hex bytes → ASCII plain text.

Commands / Tools used:

```
python3 - << 'PY'
import urllib.parse
s='47%2065%2065%202c%2020%2042%2072%2061%2069%206e%202c%2020%2077%2068%2061%20
74%2020%2064%206f%2020%2079%206f%2075%2020%2077%2061%206e%2074%2020%2074%206f%
2020%2064%206f%2020%2074%206f%206e%2069%2067%2068%2074%203f'
hexline=urllib.parse.unquote(s)
print(bytes.fromhex(hexline.replace(' ', '')).decode())
PY
```

Challenge 3

Question:

```
00111001 00100000 00110011 00110011 00111001 00100000 00110011 00110010
00110000 00111000 00110101 00111000 00110001 00110011 00111000 00110100
00110110 00110000 00111001
```

Answer:

9 339 3208581384609

How it was solved / Type of encoding:

Binary octets → ASCII. The result appears to be a numeric string; instructor may expect a second stage (e.g., treat as decimal ASCII codes).

Commands / Tools used:

```
python3 - << 'PY'
bits='00111001 00100000 00110011 00110011 00111001 00100000 00110011 00110010
00110000 00111000 00110101 00111000 00110001 00110011 00111000 00110100
00110110 00110000 00111001'
print(''.join(chr(int(b,2)) for b in bits.split()))
PY
```

Notes:

If a phrase is expected, try treating each space-separated number as an ASCII code, or as part of a phone-like number.

Challenge 4

Question:

```
48 48 49 49 49 48 48 48 ... [truncated here in doc – use the assignment’s full
line]
```

Answer:

Sugar, spice, and everything nice!

How it was solved / Type of encoding:

Step 1: Interpret numbers as ASCII to get a string of '0'/'1' groupings separated by spaces.
Step 2: Convert each 8-bit binary to a decimal ASCII code string.
Step 3: Convert those decimals to final ASCII text.

Commands / Tools used:

```
# Sketch of the pipeline
python3 - << 'PY'
nums=list(map(int, open('challenge4.txt').read().split()))
s=''.join(chr(n) for n in nums)                # step 1 -> '01001000
011001...' etc.
out=''.join(chr(int(b,2)) for b in s.split())   # step 2 -> '083 117 103 ...'
print(''.join(chr(int(x)) for x in out.split())) # step 3
PY
```

Challenge 5

Question:

```
01001110 01010111 01001110 01000110 ... 00111101
```

Answer:

With great power there must also come — great responsibility.

How it was solved / Type of encoding:

Binary octets → ASCII (Base64) → Base64 decode → Base58 decode.

Commands / Tools used:

```
python3 - << 'PY'
import base64
```

```
b='01001110 01010111 01001110 01000110 ... 00111101'  # use the full string
s=''.join(chr(int(x,2)) for x in b.split())
b64 = base64.b64decode(s).decode()
alphabet='123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz'
n=0
for ch in b64:
    n = n*58 + alphabet.index(ch)
print(n.to_bytes((n.bit_length()+7)//8,'big').decode())
PY
```

Challenge 6

Question:

```
(92E C@==D 5@H?DE2:CD[ 2=@?6 @C :? A2:CD[ C@==D @G6C J@FC ?6:893@CVD 5@8n
```

Answer:

THIS LOOKS LIKE ROT47; IT DECRYPTS TO: FROM SHERLOCK: “DID YOU MISS ME?” ;)

How it was solved / Type of encoding:

ROT47 substitution over printable ASCII (shift by 47 within 33–126).

Commands / Tools used:

```
python3 - << 'PY'
s='(92E C@==D 5@H?DE2:CD[ 2=@?6 @C :? A2:CD[ C@==D @G6C J@FC ?6:893@CVD 5@8n'
out=''.join(chr(33+((ord(c)-33+47)%94)) if 33<=ord(c)<=126 else c for c in s)
print(out)
PY
```

Challenge 7

Question:

```
Z bl gwn pzotnk dhrjatn gwbg unhfd bg vrja oztwglband!
```

Answer:

[Caesar (ROT) – use brute force below to capture the readable sentence]

How it was solved / Type of encoding:

Caesar/ROT substitution. Try all 26 shifts and pick the English-looking one.

Commands / Tools used:

```
python3 - << 'PY'
s='Z bl gwn pzotnk dhrjatn gwbg unhfd bg vrja oztwglband!'
```

```
def caesar(t,k):
    o=''
    for ch in t:
        if 'a'<=ch<='z': o+=chr((ord(ch)-97+k)%26+97)
        elif 'A'<=ch<='Z': o+=chr((ord(ch)-65+k)%26+65)
        else: o+=ch
    return o
for k in range(26):
    print(k, caesar(s,k))
PY
```

Challenge 8

Question:

```
[/]'--_|\|/ZL!\|/--. '-- <{E 3[]A<_V_'--ZLD []Z E>- _]<[|--\|/V\[-- '--
Z>\|/Z[--'--[]Z--.
```

Answer:

[Likely ROT47 or symbol-leetspeak mapping; decode with CyberChef: ROT47 / From Base (various) / Leetspeak]

How it was solved / Type of encoding:

Start with ROT47; if not readable, try translating symbol-to-letter (leet) or removing decoration, then apply Caesar/Atbash.

Commands / Tools used:

```
# Try ROT47 first
python3 - << 'PY'
s="[/]'--_|\\|/ZL!\\|/--. '-- <{E 3[]A<_V_'--ZLD []Z E>- _]<[|--\\|/V\\[-- '--
Z>\\|/Z[--'--[]Z--."
rot=lambda t: ''.join(chr(33+((ord(c)-33+47)%94)) if 33<=ord(c)<=126 else c
for c in t)
print(rot(s))
PY
```

Challenge 9

Question:

```
V3C10V1C18V3C9C3V1C11V3C10V1C9V3C10C11V4C16C10V2V1C11C16C16V4C1V2C8V2C12C16V3C
11V1C2V1C5V2
```

Answer:

[Polybius / coordinates cipher – decode with a 4x? matrix mapping V=row, C=column; include screenshot.]

How it was solved / Type of encoding:

Treat as coordinate pairs V<number> C<number>. Build a grid (e.g., 4x? or custom) and map to letters (common: Polybius square).

Commands / Tools used:

```
# Pseudocode for Polybius-style mapping (adjust size & alphabet to fit)
# Use CyberChef: 'From Polybius Square' and choose appropriate grid.
```

Challenge 10

Question:

```
9;';13;0;3;21;20;5;!;0;9;';13;0;3;21;20;5;!;0;9;';13;0;3;21;20;5;18;0;20;8;1;1
4;0;1;0;3;1;20;,;0;1;14;4;0;1;0;13;15;21;19;5;,;0;1;14;4;0;1;0;20;18;5;5;0;6;2
1;12;12;0;15;6;0;14;21;20;19;!
```

Answer:

[Likely keyboard/phone T9 or punctuation-as-word-separator mapping; decode and include screenshot.]

How it was solved / Type of encoding:

Try splitting on ';' to get numbers; punctuation (',', '!', quotes) likely denote word boundaries. Map 1–26 → A–Z.

Commands / Tools used:

```
python3 - << 'PY'
s='9;';'13;0;3;21;20;5;!;0;9;';'13;0;3;21;20;5;!;...'
# Clean and map numbers 1-26->A-Z; treat 0 or punctuation as spaces.
PY
```

Screenshots: Paste output screenshots for each challenge below this line.