

Question: Based

copy the text file to cyberchef and put magic recipe and you got the flag

The screenshot shows the CyberChef web application. On the left, the 'Recipe' panel is active, displaying the 'Magic' recipe. The 'Depth' is set to 3, and the 'Intensive mode' and 'Extensive language support' checkboxes are unchecked. The 'Crib (known plaintext string or regex)' field is empty. The 'Input' panel on the right contains a long Base64-encoded string. Below the input, a 'Decipher/strip' button is visible. At the bottom, the 'Output' panel shows a table with the results of the recipe.

Recipe (click to load)	Result snippet	Properties
<code>From_Base64({'A-Za-z0-9+/='},true,false)</code>	STOUTCTF{h55DT3c12bDy7sqJoH7qQs7dXrE4Ysd7}	Matching ops: From Base65 Valid UTF8 Entropy: 4.66

Flag: STOUTCTF{h55DTJc12bDy7sqJoH7qQs7dXrE4Ysd7}

Question: V

v.txt

Use the Giovan alphabet provided to you ABCDEFGHIJKLMNOPQRSTUVWXYZ

Can you solve the great GIOVAN mystery?

YBCPTPZN{EaT8CK2zVexEqjdCmP6URd14xW6kNg7B}

So I was searching for Giovan on chatgpt and it told me about vigenere cipher so I tried it using dcode

The screenshot shows the DCode Vigenere Cipher Decoder tool. On the left, a search bar labeled 'Search for a tool' contains the text 'e.g. type \'boolean\''. Below it, a 'Results' section shows 'Vigenere' and 'GIOVAN' (Alphabet: 26: ABCDEFGHIJKLMNOPQRSTUVWXYZ). A flag is displayed: STOUTCTF{Qft8PE2rHjxRkbpHmC6OJp14cW6xHy7N}. On the right, the 'VIGENERE CIPHER' section is active, showing 'VIGENERE DECODER'. The 'VIGENERE CIPHERTEXT' field contains 'YBCPTPZN{EaT8CK2zVexEqjdCmP6URd14xW6kNg7B}'. The 'PARAMETERS' section shows 'PLAINTEXT LANGUAGE' set to 'English' and 'ALPHABET' set to 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'. The 'DECRYPTION METHOD' section has 'KNOWING THE KEY/PASSWORD: GIOVAN' selected. A 'DECRYPT' button is at the bottom.

Flag: STOUTCTF{Qft8PE2rHjxRkbpHmC6OJp14cW6xHy7N}

Question: Jeans

Jeans.txt

```
TGATAGTGTGATTAGTCATAGGCTACGTATGCTTAGAGGGAGCTAGCGCACCGTTGCAGTCAC
TCGCATGAGCGTACATCAATTTGTTGCGAGTCTAGATCAATGATTAGTCGTGACACCCTCACG
```

Jeans sounds like gene ?? so I went to find a DNA decoder online and tried it and got the flag

DNA Writer

Translate words into a sequence of DNA bases and convert them back again.

Translate text to base sequence

Enter text to translate: or

☐ Include non-coding DNA:

Show Color Sequence: ☐ ()

Output:

Translate base sequence to text

Enter Sequence:

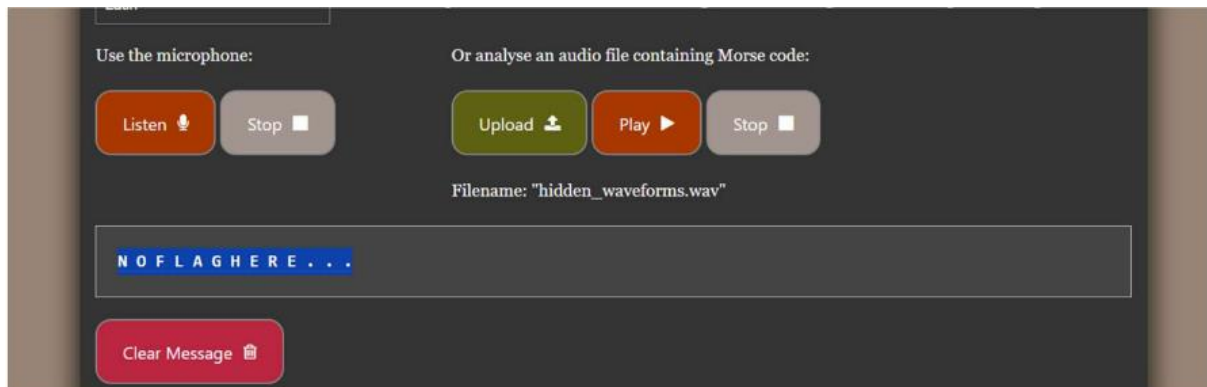
☐ Show Color Sequence: ☐

Output:

STOUTCTF.QFT8PE2RHJXRKBPHMC6OJP14CW6XHY7N.

Flag: STOUTCTF{QFT8PE2RHJXRKBPHMC6OJP14CW6XHY7N}

Question: Hidden Waveforms



First I hear the sound first and looks like morse code. But what I got is NOFLAGHERE...

This form decodes the payload that was hidden in a JPEG image or a W

Select a JPEG, WAV, or AU file to decode:

Choose File hidden_waveforms.wav

Password (may be blank):

- ☒ View raw output as MIME-type text/plain
- ☐ Guess the payload
- ☐ Prompt to save (you must guess the file type yourself.)

Submit

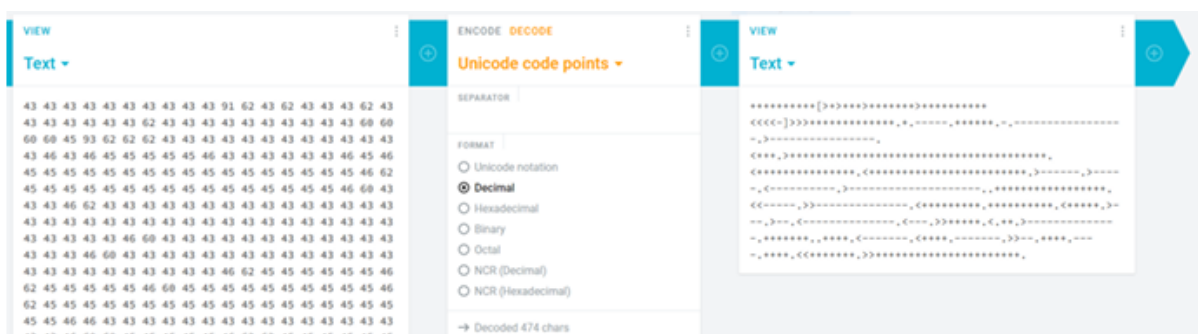
To use this form, you must first [encode a file](#).

These pages use the [steghide](#) program to perform steganography, and t

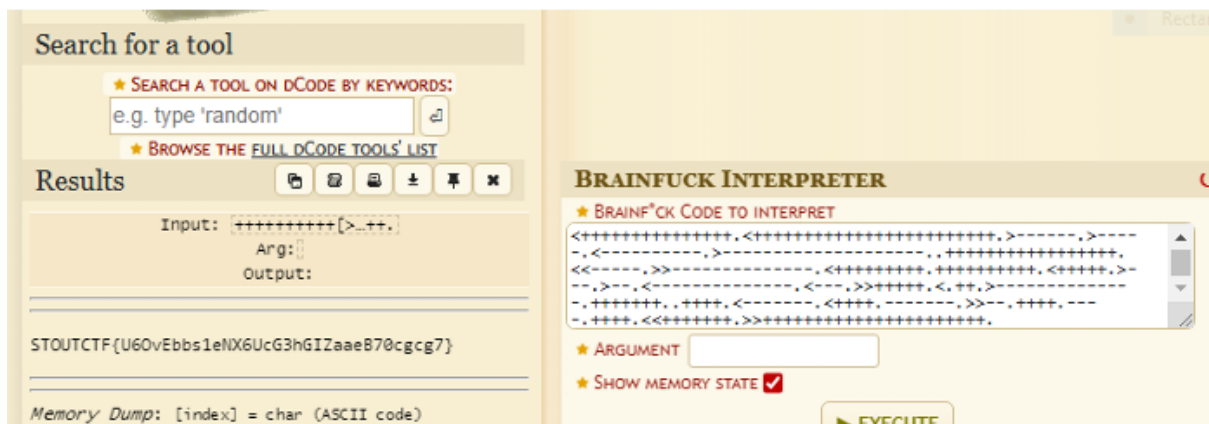
Then I try use stega tools online to extract any hidden strings or anything



This is what I got



After that I go decode this decimal



it was brainfuck cipher.and I used dcode to copy paste and decode it.

Flag: STOUTCTF{U6OvEbbs1eNX6UcG3hGIzaaeB70cgcg7}

Question: Custom Cipher

Your scripting cant break my encoding!

VWRXWFWI{Vr3J8NJks4Tn58PjDv3IPNc9VueGFwlu}



Go to Caesar Cipher in Dcode and paste the encoded flag and you'll get the flag

Flag= STOUTCTF{So0G5KGhp1Qk25MgAs0FMK96SrbDCtir}

Question: Fat Finger

DYPIYVYG}fyYV0[s-KhuDwV[OKn:Y<H:4k8CsYeFY|

I googled what is fat finger

A "fat-finger" error, or "fat-fingering," is a slang term for a typing mistake caused by accidentally pressing the wrong key on a keyboard or touchscreen, often due to keys being too close together

so I rearrange it to get the flag

flag= STOUTCTF{dtTC9pa0JgySqCpIJbLTMGL3j7XaTwDT}

Question: Nothing To See Here !

I have been given an empty file but when I can hover over it so after researching for awhile it was a whitespace language

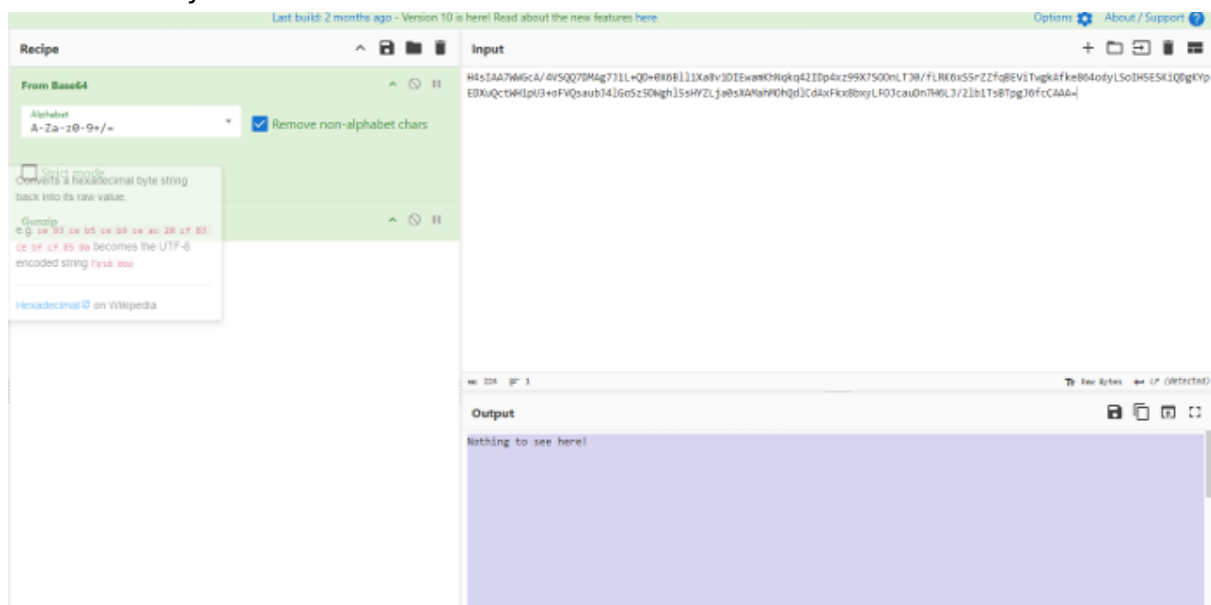
I went to dcode and paste it



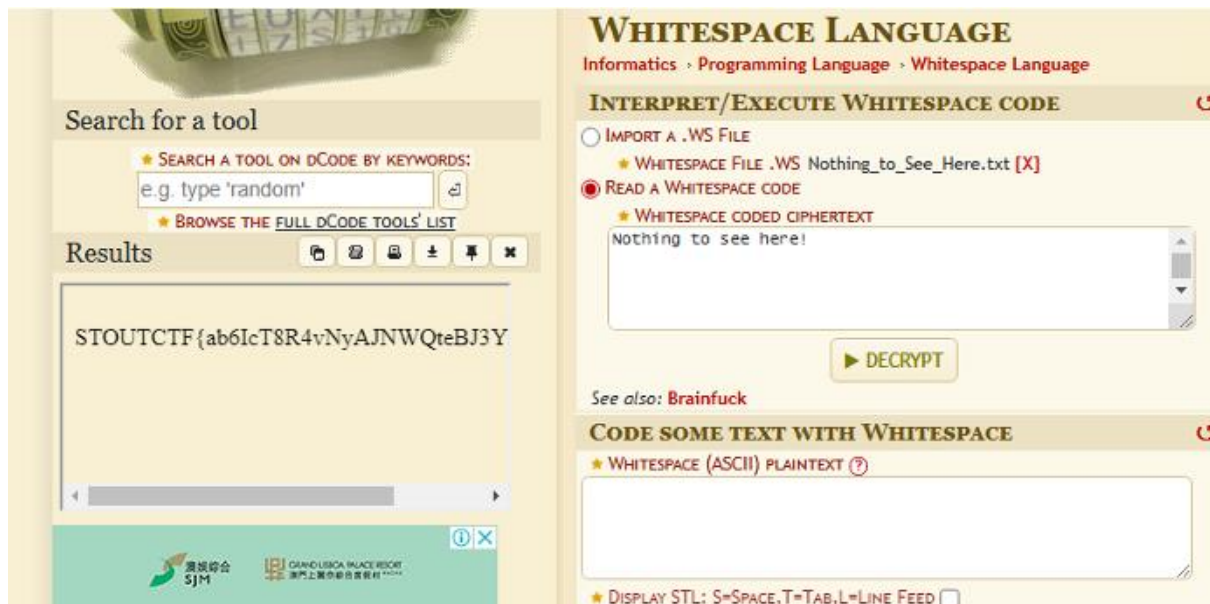
I got a based64

H4sIAA7WWGcA/4VSQQ7DMAg7J1L+QD+0X6Bl1Xa8v1DIEwamKhNqkq42IDp4xz99X7
SOOnLTJ0/fLRK6xS5rZZfqBEViTwgkAfke864odyLSolH5ESKiQDgKYpEDXuQctWH1pU3+o
FVQsaubJ4lGoSz5DWghl5sHYZLja0sXAMahMOhQdlCdAxFkx8bxyLFOJcauDn7H6LJ/2lb
1TsBTpgJ6fcCAAA=

so I went to cyberchef to decode it



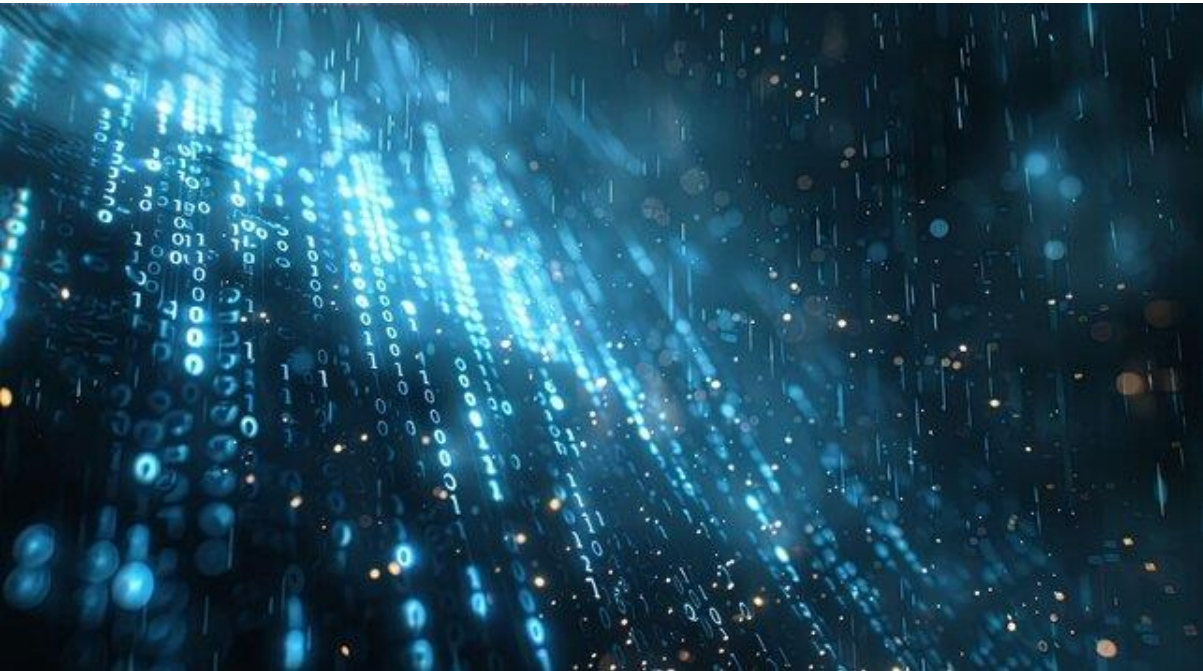
Oops it says that Nothing to see here but oh well I tried to hover it again and I cld hover over it again so I used the white space language on dcode again.



And I got the flag !

flag= STOUTCTF{ab6lcT8R4vNyAJNWQteBJ3Yd2VTrkVCp}

Question: 7 Bit Flow



We got a picture

For this challenge all u need to do is extract the data using red and choose 7 in this website (<https://georgeom.net/StegOnline/extract>) and you'll get the flag

Extract Data

Here you can extract data hidden inside of the image. Select some bits and adjust the settings appropriately. The final extracted data is checked against some basic file headers, and so the filetype can be automatically determined.

Please note that Alpha options are only available if the image contains transparency.

	R	G	B
7	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
6	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
0	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Pixel Order

Row

Bit Order

MSB

Bit Plane Order

R G B

Trim Trailing Bits

No

Go

Results

No file types identified.

The results below only show the first 2500 bytes. Select "Download" to obtain the full data.

Ascii (readable only):

STOUTCTF {7jDcMSX 51qCCM6w xr71j1TS UXRz0L11 U}.....

.....

.....

Flag= STOUTCTF {7jDcMSX51qCCM6wxr7ijiTSUXRz0LiiU}

Question: 13RottenTermites

I got a loooooong text file so I went to cyberchef

```
dXVKYmx2QW9mY1F3dEYvYm0rKzZUcURjV3M3Mkw2VENLRzBPakd3VjBRb
EFHUC9ieHp5a3RqbTVn
R2xmL0NocnRPeUHRUZdPSHp1RApLeFF4T29HQ31wN1FrNjZrdUk2bw1sR
05YR1FKV3hXQjJ5RENm
SGdNc1lxN01oZ291Zm02ZmF2eG5rbm5pY25nNU80YmZaUzN1MUxUCmJuS
zRZUC9PNU84eEd1Zj1G
dnhyTnZ3T1U1bFdCUWdYUjFBcFN3R2YydHhZQ1o4MDU3ZTBGUnBJTzBIM
EhiR25oejQ3aWJFZXR4
eEKKwndldHM5TVVYL0xtMnVmTzF5Y2ticG5DajF3bm14aVJDb1duYnVXN
UZHajQxRERScXV3QWVv
UUNZSjMvaFB10XFDVU1sQ0RyVVU0aQp4NWxpdDVoZFdwT1IrV3BGM3YxO
W9ORHJHNU9vUFRSd21l
R1pKNnI2djC2WwEXRXRVb2ZEVHBvMk1UNld5c2V0NzZwRFYwZmNHekxKC
mVNNzRwbN2YzJ1aGx3
VnBXbzFicEdTaEdYdFdBTERTZbDRkRGtnRWN3aFIyeVg3WVFzVVBIM1dVo
```

I went from base64 to decrypt to rot13 because it's the hint from the challenge name and I found the Flag hidden in the long text files.

The screenshot shows the CyberChef web application interface. On the left, the 'Recipe' panel is configured with 'From Base64' and 'ROT13' operations. The 'ROT13' operation has 'Rotate lower case chars' and 'Rotate upper case chars' checked. The 'Input' panel on the right contains a long text file. The 'Output' panel at the bottom displays the decrypted result, which is a long string of characters. The flag 'STOUTCTF{qKp1MOJMDaJUII5KybLrcfZOFQ3IN1j2}' is highlighted in the output.

Flag= STOUTCTF{qKp1MOJMDaJUII5KybLrcfZOFQ3IN1j2}

[illegible]

solve.py

```

from heapq import heappush, heappop

class Node:
    def __init__(self, symbol, freq): # Corrected constructor method name
        self.symbol = symbol
        self.freq = freq
        self.left = None
        self.right = None

    def __lt__(self, other):
        return self.freq < other.freq

def build_huffman_tree(probabilities):
    pq = []
    for symbol, freq in probabilities.items():
        heappush(pq, Node(symbol, freq))

    if len(pq) == 1:
        return pq[0] # Handle the case with a single type of symbol

    while len(pq) > 1:
        left = heappop(pq)
        right = heappop(pq)
        merged = Node(None, left.freq + right.freq)
        merged.left = left
        merged.right = right
        heappush(pq, merged)
        print(f"Merged nodes: ({left.symbol}, {left.freq}) + ({right.symbol}, {right.freq})")

    return pq[0] if pq else None

def generate_codes(node, prefix="", code_map={}):
    if node is not None:
        if node.symbol is not None:
            code_map[node.symbol] = prefix
            generate_codes(node.left, prefix + "0", code_map)
            generate_codes(node.right, prefix + "1", code_map)
    return code_map

def decode_huffman(encoded, code_map):
    reverse_map = {v: k for k, v in code_map.items()}
    current_code = ""
    decoded_message = ""

    for bit in encoded:

```

```

current_code += bit
if current_code in reverse_map:
    decoded_message += reverse_map[current_code]
    current_code = "" # Reset the current code after decoding
return decoded_message

# Example probabilities dictionary
probabilities = {'co': 0.007352941176470588, 'me': 0.007352941176470588, 'e': 0.01838235294117647, 't':
0.01838235294117647, 'to': 0.011029411764705883, 'o': 0.011029411764705883, 'ou': 0.01838235294117647, 'ut':
0.007352941176470588, 's': 0.025735294117647058, 'CT': 0.007352941176470588, 'TF': 0.007352941176470588, 'l':
0.011029411764705883, 'l': 0.007352941176470588, 'm': 0.007352941176470588, 'm': 0.011029411764705883, 's':
0.007352941176470588, 'h': 0.007352941176470588, 'ha': 0.007352941176470588, 'y': 0.007352941176470588, 'y':
0.007352941176470588, 'yo': 0.007352941176470588, 'w': 0.007352941176470588, 'er': 0.007352941176470588, 're':
0.011029411764705883, 'a': 0.014705882352941176, 'ab': 0.007352941176470588, 'le': 0.007352941176470588, 't':
0.022058823529411766, 'th': 0.014705882352941176, 'hi': 0.007352941176470588, 'is': 0.011029411764705883, 'm':
0.007352941176470588, 'es': 0.007352941176470588, 'ss': 0.007352941176470588, 'ag': 0.007352941176470588, 'e':
0.007352941176470588, ' ': 0.011029411764705883, 'as': 0.011029411764705883, 'i': 0.014705882352941176, 'it':
0.011029411764705883, 'ar': 0.007352941176470588, 'ur': 0.007352941176470588, 'ne': 0.007352941176470588, 'd':
0.007352941176470588, 'o': 0.007352941176470588, 'on': 0.007352941176470588, 'c': 0.007352941176470588, 'la':
0.007352941176470588, 'wa': 0.007352941176470588, ' ': 0.007352941176470588, 'W': 0.022058823529411766, 'e':
0.058823529411764705, 'l': 0.025735294117647058, 'c': 0.01838235294117647, 'o': 0.058823529411764705, 'm':
0.022058823529411766, ' ': 0.13970588235294118, 't': 0.05514705882352941, 'U': 0.007352941176470588, ' ':
0.003676470588235294, 'S': 0.007352941176470588, 'u': 0.022058823529411766, ' ': 0.011029411764705883, 's':
0.05514705882352941, 'C': 0.011029411764705883, 'T': 0.01838235294117647, 'F': 0.007352941176470588, 'l':
0.007352941176470588, 'l': 0.011029411764705883, 'h': 0.025735294117647058, 'a': 0.051470588235294115, 'p':
0.014705882352941176, 'y': 0.029411764705882353, 'w': 0.011029411764705883, 'r': 0.03308823529411765, 'b':
0.014705882352941176, 'd': 0.014705882352941176, 'i': 0.025735294117647058, 'g': 0.01838235294117647, ' ':
0.022058823529411766, '?': 0.003676470588235294, 'n': 0.025735294117647058, 'f': 0.007352941176470588, 'A':
0.007352941176470588, 'H': 0.003676470588235294, ' ': 0.003676470588235294, 'O': 0.003676470588235294, 'l':
0.003676470588235294, 'o': 0.007352941176470588, 'L': 0.003676470588235294, 'Z': 0.003676470588235294, 'v':
0.003676470588235294, 'E': 0.003676470588235294, '2': 0.003676470588235294, '3': 0.003676470588235294, 'N':
0.007352941176470588, 'K': 0.003676470588235294, 'k': 0.003676470588235294, '8': 0.007352941176470588, 'J':
0.007352941176470588, '6': 0.007352941176470588, 'M': 0.003676470588235294, 'x': 0.003676470588235294, '7':
0.003676470588235294, 'j': 0.003676470588235294}

# Actual Huffman encoded message
encoded_message =
"0111011110110011111001110110010100110000110100000100101110111110100111101110111101000111100001010100
011001011010111000001010110101110111111101011010000011000011000011000001111011111011001110010011
1011000010010110100111100110101100001101001101101111010110010001111000001101000111101110010111100000
000000001101011111111111100001011101011111110000100011000011001000110110111110111011010111101111
00110100111000100011101101111111100011010100111001101000111010111110000001110100110100111000111
1011100101111111010100111010100110000100001111010111010000111001101010011101001011110110100101111
10010000000000111010110000101000001000000001111010101001000001110111001001101010010111010001101111
011001001011111101111001100011000100111110011110000011010101101001111000000010111100101100110110110
01000011011000010010010111111110000100110100000111110010001010010100001000011101110101010001000100
01001010101110010110101110110010001110101010001011110011000110011001010101111010111001101011011011
110111110011001101010001010111000111011111010111010000011100101111001001111000111011110010011100101
011101000101111000111010111101011011010101000100011011010111101010011010100100010111010001011111
101000100010011111011110011000110001001111100111100000110101011010011111000000010111100101100110110110
01000011011000010010010111111110000100110100000111110010001010010100001000011101110101010001000100
01001010101110010110101110110010001110101010001011110011000110011001010101111010111001101011011011
110111110011001101010001010111000111011111010111010000011100101111001001111000111011110010011100101
01110100010111100011101011110101101111010110100010001101101011110101001101010010001011101000101111
101000100010011010101101100000101"

# Running the Huffman Tree functions
root = build_huffman_tree(probabilities)
if root:
    code_map = generate_codes(root)
    print("Code Map:", code_map) # Output the Huffman codes for review
    decoded_message = decode_huffman(encoded_message, code_map)
    print("Decoded Message:", decoded_message) # Print the decoded message
else:
    print("Failed to build Huffman tree.")

```

Output of the script

Decoded Message: Welcome to UW-Stout's CTF! I'm so happy you were able to decrypt this message. Was it hard? I'm not sure. I learned about this algorithm in one of my classes and thought it was cool...Anyways. Here is your flag:

STOUTCTF{A0LZTvEW23NcbeKk8JyWJ8W0b6Mx7p6N} Congrats!

Flag: STOUTCTF{A0LZTvEW23NcbeKk8JyWJ8W0b6Mx7p6N}