37. Write a python program that can perform a letter frequency attack on any monoalphabetic substitution cipher without human intervention. Your software should produce possible plaintexts in rough order of likelihood. It would be good if your user interface allowed the user to specify “give me the top 10 possible plaintexts.”

Code:

```
import string
from collections import Counter
import itertools
# English letter frequency ranking (most → least common)
english_freq_order = "ETAOINSHRDLCUMWFGYPBVKJXQZ"
def frequency_attack(ciphertext, top_n=10):
    # Count frequency of letters in the ciphertext
    text = ciphertext.upper()
    letters_only = [ch for ch in text if ch in string.ascii_uppercase]
    freq = Counter(letters_only)
   # Sort ciphertext letters by frequency (highest → lowest)
    cipher_freq_order = ''.join([x for x, _ in freq.most_common()])
    # We try several permutations of the top few letters to improve accuracy
    # For simplicity, permute top 6 English letters
    base_map = list(english_freq_order[:6])
    guesses = []
    # Try permutations to produce multiple likely plaintexts
    for perm in itertools.islice(itertools.permutations(base_map), top_n):
        # Create substitution map
        mapping = {}
        for i, ch in enumerate(cipher_freq_order[:len(perm)]):
            mapping[ch] = perm[i]
```

```
        # Fill remaining letters by frequency order
        remaining_plain = [c for c in english_freq_order if c not in perm]
        remaining_cipher = [c for c in cipher_freq_order if c not in mapping]
        for c, p in zip(remaining_cipher, remaining_plain):
            mapping[c] = p
        # Apply mapping
        plaintext = ""
        for ch in text:
            if ch in mapping:
                plaintext += mapping[ch]
            else:
                plaintext += ch
        guesses.append(plaintext)
    return guesses
# -------------------------------------------------
# Example usage
# -------------------------------------------------
cipher = input("Enter the monoalphabetic ciphertext:\n")
n = int(input("How many top plaintext guesses do you want? "))
results = frequency_attack(cipher, n)
print("\nTop", n, "possible plaintexts:\n")
for i, guess in enumerate(results, 1):
    print(f"{i}. {guess}")
```

```
>>>
========== RESTART: C:/Users/Maria/OneDrive/Documents/ex37.py ==========
Enter the monoalphabetic ciphertext:
Attack
How many top plaintext guesses do you want? 2

Top 2 possible plaintexts:

1. ETTEAO
2. ETTEAO
>>>
```