```
from typing import List, Set
from difflib import get_close_matches

# QWERTY Braille to character mapping (sample subset)
BRAILLE_MAP = {
    frozenset(['D']): 'A',
    frozenset(['D', 'W']): 'B',
    frozenset(['D', 'K']): 'C',
    frozenset(['D', 'Q']): 'E',
    frozenset(['D', 'K', 'W']): 'G',
    frozenset(['W', 'Q']): 'I',
    frozenset(['W', 'Q', 'K']): 'J',
    frozenset(['D', 'W', 'K']): 'F',
    frozenset(['D', 'W', 'P']): 'H',
    # ... add more mappings as needed
}

# Sample dictionary
DICTIONARY = ['cat', 'cab', 'bat', 'rat', 'chat', 'mat', 'cart']

def braille_input_to_text(input_sequence: List[Set[str]]) -> str:
    decoded = ''
    for braille_char in input_sequence:
        char = BRAILLE_MAP.get(frozenset(braille_char))
        if char:
            decoded += char.lower()
        else:
            decoded += '?'
    return decoded

# Simple Levenshtein Distance implementation
def levenshtein_distance(a: str, b: str) -> int:
    m, n = len(a), len(b)
    dp = [[0] * (n+1) for _ in range(m+1)]

    for i in range(m+1):
        dp[i][0] = i
    for j in range(n+1):
        dp[0][j] = j

    for i in range(1, m+1):
        for j in range(1, n+1):
            cost = 0 if a[i-1] == b[j-1] else 1
            dp[i][j] = min(dp[i-1][j] + 1,       # deletion
                           dp[i][j-1] + 1,       # insertion
                           dp[i-1][j-1] + cost)  # substitution
    return dp[m][n]
```

```
# Get suggestions using Levenshtein distance
def suggest_corrections(decoded: str, dictionary: List[str], max_suggestions=3) -> List[str]:
    scored_words = [(word, levenshtein_distance(decoded, word)) for word in dictionary]
    scored_words.sort(key=lambda x: x[1])
    return [word for word, _ in scored_words[:max_suggestions]]

# Full system integration
def autocorrect_braille_input(input_sequence: List[Set[str]]):
    print("Raw Input Sequence:", input_sequence)
    decoded = braille_input_to_text(input_sequence)
    print("Decoded Word:", decoded)
    suggestions = suggest_corrections(decoded, DICTIONARY)
    print("Suggestions:", suggestions)
    return suggestions

# ----- Sample Test -----
if __name__ == "__main__":
    # Simulated user Braille input: 'C', 'A', 'B'
    sample_input = [{'D', 'K'}, {'D'}, {'D', 'W'}]
    autocorrect_braille_input(sample_input)
```