Assignment: Compression with Error Detection Using Huffman Coding

Objective

- Implement **Huffman coding** for text compression.
- Introduce a basic **error detection mechanism** to ensure the integrity of compressed data.
- Analyze the efficiency of the compression and the effectiveness of the error detection.

Instructions

1. **Implement Huffman Encoding**
    - Write a function that generates a **Huffman tree** for a given input string based on character frequencies.
    - Encode each character in the string using the Huffman tree to create variable-length binary codes, with more frequent characters having shorter codes.
    - Compress the input by replacing each character with its corresponding binary code.
    - Test your function with text data of varying lengths and character distributions to see how Huffman coding adapts to different frequencies.
2. **Implement Huffman Decoding**
    - Write a corresponding decoding function that takes the binary-encoded data and the Huffman tree to reconstruct the original text.
    - Verify that the decompressed output matches the original input exactly.
    - Test your decoding function using the same inputs as the encoding function to ensure correctness.
3. **Error Detection with Checksum**
    - Add a basic error detection mechanism:
        - Implement a **checksum** (e.g., a simple sum of ASCII values of the original characters) and store it with the compressed data.
    - Modify your decoding function to:
        - Recalculate the checksum after decompression.
        - Compare it to the stored checksum to confirm data integrity.
        - Raise an error or alert if the checksums do not match, indicating potential data corruption.
4. **Analysis and Reflection**
    - **Compression Efficiency**:
        - Calculate the **compression ratio** for each test case by comparing the sizes of the original and compressed data.
        - Discuss how the distribution of character frequencies affects the compression ratio and where Huffman coding might be most effective.
    - **Error Detection**:
        - Reflect on the effectiveness of the checksum approach in detecting errors in compressed data.

- Briefly discuss how a more advanced error detection technique could improve this process, without requiring full implementation.

Extra Credit

- **Alternate Error Detection**: Implement a **Cyclic Redundancy Check (CRC)** for more robust error detection, and compare its effectiveness to the checksum.
- **Adaptive Encoding**: Experiment with resetting the Huffman tree for portions of long texts to adapt the encoding to changing character frequencies. Analyze any changes in compression efficiency.

Submission Guidelines

- **Code Files**: Submit your source code files for the Huffman encoding, decoding, and error detection functions.
- **Analysis Document**: Include a report with compression ratio calculations, reflections on error detection, and answers to analysis questions.