**Kristina Rizzo -- Read Me File**

In this lab we were assigned to implement a Huffman encoding and decoding system to compress and decompress textual data.

I used Visual Studio Code to develop this project. When running the program make sure to have the latest version of Python installed as well as make sure that all of the proper files are located in the same directory, run the main program using python main.py, and make sure to check the output results. An important note is that for larger input texts, you may experience better compression rates compared to smaller inputs due to the frequency distribution characteristics of natural language.

The structure of the code consisted of the HuffmanTree Class, HuffmanCoder Class, ErrorHandling Class, the Main Script, and the Input/Output files.

In the HuffmanTree Class, our code builds a proper Huffman tree based on character frequencies, implements precedence rules for tie-breaking (giving single letters precedence over groups, then alphabetically), generates variable-length codes through tree traversal, and provides methods for printing the tree in preorder traversal.

In the HuffmanCoder Class, our code handles the encoding and decoding operations using the generated Huffman codes, validates input for both encoding and decoding operations, reports errors for unrecognized Huffman sequences during decoding, and filters input text to keep only relevant characters.

In the ErrorHandling Class, our code was written to validate inputs, check for binary vs. alphabetic characters, remove spaces from input, handle errors through a consistent response structure, and provide meaningful error messages.

In our Main Script, the file reads input strings from input.txt, creates an instance of the HuffmanTree class with the given frequency table, builds the tree and generates codes, processes each line for either encoding or decoding based on content type, measures execution time for all operations, calculates compression and decompression statistics, and writes comprehensive results to the output.txt file.

The Input file contains strings to encode and decode, and the Output file stores the execution results including the tree structure, codes, encoded/decoded outputs, and compression statistics.

In the future, I would like to implement additional compression algorithms to compare with Huffman encoding and potentially develop an adaptive version that updates the tree dynamically as new data arrives.