Max Chi
ECE 368 Project 2 Milestone 1

The purpose of project 2 for ECE 368 is to understand and implement Huffman Coding into compressing a given text file through manipulation at the bit level. In trying to meet the project goal of this assignment, it is necessary to first understand the significance of ASCII coding, in that each character, number, symbol, etc… can be represented in ASCII code as well as broken down further into bits, in this case we focus on 8-bit binary values. By first breaking down the text file and keeping track of the frequencies of variable, we then sort them into a coding tree in which they would be organized based on reducing the ASCII values (or binary) to help with mapping within the newly created tree. The organizational process utilizes Huffman coding by counting the frequencies of each symbol and then giving them a "weight" to determine the order in which they would be sorted into the tree. During sorting, to make the information easier to manage, a table will be created to store the symbols along with their frequencies. The table helps in organizing the frequencies and their respective symbols in an attempt to help compress or lessen the amount of bits required to access a symbol. At the same time, the Huffman code can be used to help organize one side of a binary tree. Based on the given example "go go gophers" I have come to an understanding on how it will be convenient to handle the process of converting letters to ASCII code first, and then taking the corresponding 8-bit binary values of each ASCII code value. This seems to somewhat simplify the transition of keeping up with what each string of bits represents. I have an idea of how to compress the bits of each character, but still do not quite understand how to reinterpret the characters into the corresponding symbols/ values when going through the decompressing /interpretation process, however that will be dealt with later. After we evaluate the frequencies of letters / symbols /characters used, it will be easy to compress or use less bytes based on how many unique variables appear in the sorted table. Taking the earlier "go go gophers" example, it is clear that there are 8 unique character/ symbols used and we can assign them new bit values as long as we keep track of how they are being referred to as. In this case, we case utilize the 3-bit binary values since 2^3 covers 8 different possibilities, exactly covering the necessary character requirements. This transition from 8-bits to 3-bits to designating the character/ symbols will effectively the bits used significantly. Now transitioning to the tree, each unique value will be sorted into a binary tree and will be accessed by either traversing down the 0's or 1's edge, (0) representing the left and (1) representing the right side of a node. By doing this, the most frequently used characters / symbols will require less bits to access, while the less frequent ones might utilize larger strings of bits to access, but based on their frequencies will still be more convenient. At this point I really am a bit concerned about one part, namely the "unhuff" part of the assignment, where we take the binary file and redirect, effectively transferring the newly compressed information into a new text file. I am not sure how exactly when reading the file will we be able to distinguish what size of bits each character / symbol requires. Since the more frequent ones will require less bits to access, while the opposite for the less frequent characters. Otherwise, I think I have the general grasp of the project.