ECE 368 Milestone #2
Ranjan Behl

This project asks us to write two different programs. One is titled as huff.c and the other is unhuff.c. Both take in an input file, huff.c takes in an uncompressed file and unhuff.c takes in a compressed file. Huff.c compresses the given input and writes the header info that contains the Huffman encoded binary tree and as well the encoded text to a file that has a extension of .huff, while unhuff.c decompresses the given .huff file by reading the header info and reconstructing the tree and decoding the given encoded text and writes the uncompressed data to a file ending in .unhuff(this extension is added to the end of the given file's extension), this file must match the first file read by the huff.c program. These two programs do the opposite of each other.

Since it is given that the input file will be always be a pure ascii text file. The first function, the huff, must convert the ascii text into a huffman encoded binary tree. From the binary tree I can compress the file into the actual text. For the second function, the unhuff, the input file contains the actual text and I must revert back to my tree to retrieve the huffman code for each character. From the huffman code I can then finally decompress the file into ascii text once again.

In Huffman programming, characters are assigned huffman encoded bits based on how frequently they appear in the ascii text. Characters that appear more frequently have a lower bit value than their counterparts, Based on this I can use the greedy huffman algorithm to build the tree by first creating a sorted linked list of Nodes based on weight(frequency of appearance)(each node holds the character, its frequency, left and right child), then the two minimal weighted nodes(the first two nodes) are combined to create a new Node whose weight is the sum of the two combined nodes. Then this is node is inserted into the linked list based on its weight. This combining process is repeated until there is only one element left in the linked list, this one element will be the Huffman tree. Then a Huffman table was created traveling from the root Node and traveling to every leaf node and every left branch is "0" and right branch is "1". Then before the file is encoded the tree is traveled using post order and this post order result is written to the header info. Finally, the given input file is encoded using the Huffman table thus compress and then write to the output file. Then for the unhuff.c the header info is read from the file and the tree is reconstructed and then the encoded file is read by its binary then the process mentioned above is done in reversal to decode the file and uncompressed the file.

My strategy to code this project is to look at some past ECE 264 hw that deal with Huffman programming, form which I will be able to see how to properly read the input file, convert ascii to huffman tree and finally do the reverse of that. Furthermore, I will also be able to see how to correctly build the Huffman table and the decoder.