David Pimley

ECE 368

10/12/17

ECE 368 Project 2 Milestone 1

Since the project is broken up into two parts (huff.c and unhuff.c) where each file should correctly compress and decompress a file it would make sense to break up the project into two parts. While there are some functions that could be utilized by both files the majority of the code for each of the files will be different.

For huff.c the file should be able to take in an input file and output a file that is compressed using a Huffman algorithm. Initially the file will need to be read in to be able to receive the data and also to count the frequency of each of the letters (alphanumeric or special) and store this into an array where each character frequency can be accessed in an array of 256 integers where the integer stores the amount of times a character is used in the input file. After the data is read in and stored into an array and the frequencies are recorded in a "frequency count" array the Huffman tree will be created using two different functions. The first function will create a node of all the characters used in the input file and store these node pointers in an array. This array will then be passed to the second function where it will combine all of the "forests" into one tree where the actual Huffman code, bit value of each character can be stored. The return value of this function will be a pointer to a tree, in this case the tree will be the complete Huffman tree off all the values and their respective Huffman codes. The last function in this file will be to write the bits to a file. I feel this will be the most complicated function in the file because it will not only require bit manipulation but also being able to make sure the buffer of bit values will not cut off any values from the next or previous bits that are read in. One of the more important aspects of this function will be to be able to represent the Huffman data in the header of the file. For this I will use a preorder traversal to show the tree that represents the Huffman data that was created earlier on in this file.

For unhuff.c there will have to be two things to done. The first of which will have to be reading in the input data (including the header of the compressed file) and to decompress the file into a readable file that should be the same as the file that was inputted into the huff.c function/file. The input function of this file should correctly read in the file but also create a Huffman tree that is represented by the header of the file that is passed to it. This will be one of the more complex functions of the file seeing that it will have to create a tree from a preorder traversal. After this tree is created the binary file will have to be read and output into a new file. This will yet again require a buffer since C can only write/read bytes at a time. The next function will read a byte at a time and try to convert each sequence of bits into a character to be output into a new file, if it can't convert the set of bits into a new file then it will read in another bit and see if the next byte will add enough information so that a character can be created.

The more complex of these two files will be the unhuff.c because it will need to turn "garbage data" into real values that can be output into a file.