**Implementation:**

Implementing the codebook builder was probably the longest part. That is all covered in asst2.h and asst2.c. The First thing of note in there is how we built the tree. The leaves in our tree are built in two structs. The first struct, called treeNode, contains a pointer to the leaf on the left and leaf on the right and a pointer to another struct, called fileList. FileList contains a frequency tracker called counter, the file the word was found in, and a pointer to the file that comes next. At first when we built this we tried using only one struct. This wasn’t working after a while and was slow when it did work. Somehow splitting this up into two structs and linking them together was faster and worked better. Eventually after going through the directory recursively or just the file it populates the tree with all the words. Then It passes this tree of “double structs” (the term I coined to describe the weird struct structure we built to make this work) to a minheap making function called “createMinHeap”. We did this because making the bit representations of the words in the tree was easier this way over using the tree alone. Then from there the program would assign values to the words using 0’s and 1’s and add these to the codebook. Something to note is that our codebook doesn’t print out in a specific order. For example, the word “hi” could have the highest frequency and have a bit representation of ‘0’, but if it’s the seventh word encountered it will print out seventh in the codebook. This was something we weren’t able to fix.

Implementation of compression is done in compression.c. How it works is by taking the file name/directory name and find the file/directory. After finding it, it iterates through the file/files and tokenizes them sequentially not based of frequency and generates the new file based off what words appear and using the codebook.

**ADD DECOMPRESS WHEN DONE**

**Test Cases:**

We have built several testcases. One of our testcases was passing in just an empty file. Another one was a directory containing some empty files and other populated files. One test cases also contained subdirectories with files inside them. We did build one test cases that contained two copies of Hamlet. **ADD RESULTS WHEN DONE**