1. 8)  **W Sections**: In addition to the paper from 7) above, write a second, ~2-3 page paper in which you discuss in detail your modifications to the program and how you got the streaming character input, the variable length codes and the dictionary reset to work. Explain in detail everything you did to the program and why, especially issues that caused you any grief. This paper will be evaluated for both form and content. Be sure to submit this paper as a .doc or .docx file so that we can easily insert comments and changes. You may also be required to write a revision of your paper later on.

The main part of this assignment was to modify a given LZW compression implementation provided by the textbook author. My modifications consisted of using a DLB for the codebook, streaming character input, variable length codes, and dictionary resetting.

The easiest of the four changes was to change from reading the input file all at once to reading it as a stream. I did this by using the author’s two classes BinaryStdIn.java and BinaryStdOut.java. Both of these files have methods which allow for reading of bit (boolean) data and character length words. The biggest difference between my code and the original in this case was that I had to keep track of whether or not I was at the end of the file using the isEmpty() method.

In terms of storing the codes in a codebook, I decided to modify my DLB from assignment one. A DLB is a perfect solution for such a codebook because it offers fast searching. I added some new methods (search1(), put) and modified the searchPrefix in order to optimize repetitive searching. This gave constant time add and search after appending a single character to the StringBiulder object.

The hardest parts of this project were changing to a variable length codeword and what to do during the dictionary reset. Switching from a fixed width of twelve-byte codes to a variable width codes did cause some problems. In terms of debugging, the challenge came when switching between the last code of one width to the first code of the next width and after a dictionary reset.

Using print statements, I discovered that I was always losing the last character and its associated codeword. This mean that both in compression and expansion I was off by one that that interface.

I found a solution that worked by changing where I was outputting the codeword to the file to before I did the code width expansion. I also added one to the max codeword in the compression and in the expansion to keep them in sync.