Please review the codes for both implementations and explain the difference in both implementations. Also, indicate why you will use one over the other? Please submit your explanation as a text file and it should not be more than a page.

Sol)

Code used in Lab Exercise 1 is basically implementation of the Naïve algorithm to finding frequent pairs for a given input file of values. This Naïve approach involves finding frequent pairs by reading file once and counting each pair occurrences in the main memory. Finding the count for all possible pairs and generate all possible pairs for items in the basket and update count for each pair. The main disadvantage of this is Main Memory bottleneck, this occurs as the counting is limited by main memory and swapping the counts in and out will be a critical operation. If we use a 1D array to keep track of counts of all possible pairs i.e. (n\*n-1)/2 could need around 20 GB of memory.

To overcome this bottleneck, an efficient way was demonstrated in the Lab Exercise 2. This is a A-Priori algorithm running in two passes. The main advantage of using this algorithm is it limits the need for main memory. In first pass, it requires memory proportional to number of items and in second pass it tracks count of candidate pairs, read baskets and count in main memory only those are frequent.

The map and reduce in two phases compute the frequent items in a large collection of baskets within two passes handling the main memory issues and effectively using the main memory. So, the second implementation is preferred over the naïve approach.