**Program Report**

The program has a main function which, when called, runs sequentially through every other function until it gets to the end. It is then, when it generates an output txt file which contains the results.

First, the program prompts user to specify an input txt file and applies the first chunk of data-cleaning code, which eliminates any “ ‘ “ symbol in the text imported from the input file, uppercase letters and subsequently creates a list that contains as many elements as the number of lines in the text file.

Next, I go through the list to substitute any other symbol by a space and set a one-space separation between words for those names with two or more consequent symbols. Once that job is finished, I split every name by words, creating a list of lists which contains all names imported from the input file separated by line and words.

Now everything is ready to start generating abbreviations. To do so, I define a function that calculates the value of every letter in a word according to the provided instructions. Next, I define another function that takes a list of tuples [(letter, value)…] per name and generates all three letter abbreviations and their respective value, what is finally appended to another list. So in summary, for every name, I store its letter-value pairs in a list of tuples. Then I run the function called Abbreviations on that list, which creates three-letters abbreviations and append them along their aggregate letters value to another list, which will contain as many lists as names and will store tuples that contain all the abbreviations and their final value for every name.

Next, I define and call the function RemoveDuplicates to discard abbreviations that appear more than once per name, keeping only the one with the lowest value. To do so, I use try-except-else to avoid using multiple if statements. This function’s output is a list of dictionaries containing non-duplicated abbreviations and their value per name. After that, I run CountAndDiscard, which counts how many times each abbreviation has occurred across all names (or dictionaries) and then it stores in a list of lists those abbreviations that only appear once. Finally, the program selects the lowest-score abbreviation per name from all the remaining abbreviations until that point, generating a list of lists containing the final result.

Lastly, I store the final result in names\_abbrevs.txt along with the initial names, showing each initial name and its corresponding final result on different lines.