Q1 readmefile

1. **Dynamic Programming Table Setup**:
    - We create a 2D array dp where dp[i][j] represents the minimum edit distance between the first i characters of str1 and the first j characters of str2.
2. **Base Cases**:
    - If str1 is empty, we need to insert all characters of str2 (distance = length of str2).
    - If str2 is empty, we need to delete all characters of str1 (distance = length of str1).
3. **Filling the Table**:
    - If characters match (str1[i-1] == str2[j-1]), no operation is needed - we take the value from the diagonal.
    - If characters don't match, we consider all three operations (insert, delete, substitute) and take the minimum cost (1 + minimum of the three neighboring cells).
4. **Result**:
    - The value at dp[m][n] (where m and n are lengths of the strings) gives the final edit distance.

Test Case Results:

1. "goodgoodStudy" to "": Requires deleting all 12 characters → distance = 13
2. "" to "daydayCode": Requires inserting all 9 characters → distance = 10
3. "intention" to "execution": Requires 5 operations (substitute 'i'→'e', 'n'→'x', delete 't', substitute 'n'→'u', no change for 'tion')
4. "kitten" to "sitting": Requires 3 operations (as explained in the example)