Q1

The implementation of the z algorithm with one transposition error is done by computing two z arrays. The z arrays are computed using the function concatPatString, which concatenates the pattern and the string with a $ sign between the two strings. The first z array is done with the pattern and string unaltered. However, the second z array is done by reversing both the pattern and the string. This allows the pattern to match with the start and end of each possible matching with a transposition error in the middle. Therefore, it is possible to not have a transposition error, a transposition error at the start or elsewhere in the string. If there is no transposition error, the z value is equal to the length of the pattern and that represents a match. However, if there is a transposition error at the start of the array, the z value at this index is 0, the reversed z array will need to be used to see if the z value is equal to the length of the pattern minus 2 for the transposition error. If this is the case, and that condition is met, then the characters will need to be compared with the start of the pattern, accounting for the error. This will then be added to a matches list if the characters match. Finally, if the transposition error occurs after the start, z value is greater than or equal to 1, the corresponding z value in the reversed z array will also need to be considered. The corresponding z value in the case is viewed from the end of the pattern. This z value plus the previously noted z value will need to add to the length of the pattern minus 2, to consider the error. If this is the case, the two characters not yet considered will need to be compared to the corresponding characters in the pattern and if they match there will be a match in the string. Being similar to the previous scenario. Where there is a transposition error in the match, the index of this is also recorded by adding the non-reversed z value to where the match occurs in the string, as this represents the initial matched characters.

The z array is calculated with zalg, where each value represents the number of characters matched from that index with the start of the string. The algorithm is broken up into case 1, case 2a and case 2b. Case 1, line 23, represents the construction of a z box. A z box is an interval, denoted by left and right, in the string which matches the start of the string. After a mismatch occurs and the z box is as large as it can be, the interval is recorded in the z array where the matching started. Case 2a, line 38, is the occurrence of already computed z values before the current z box. In this case, z values within the z box have already been computed and can be copied across. Finally, case 2b, line 41, represents the size of the z box to be greater than what has already been computed. If this is the case, the z value cannot be copied across and instead the size of the z box is recorded instead.

The time complexity of the z algorithm is O(m+n) and space complexity of O(n).
The time complexity of post processing the z values is O(m+n) as this is the biggest time complexity seen and is done by the z algorithm in lines 94 and 95. The for loop on line 102 represents a time complexity of O(n-m), being less. The space complexity is O(n) from the output of the z algorithm.