Question 1: Explanation

In Question1, the faction takes two strings (s and t) and find out whether one (t) is anagram of the other (s). Here, in the string t, all the different letter combinations are considered to find out anagrams. If an anagram is found, the program returns true.

The efficiency of the program depends on how long are the given strings. However, to increase the time efficiency itertools.permutations() function is used to generate different letter combinations in a word.

word\_combination\_list = list(itertools.permutations(t\_list, len(t\_list)))

This function is helpful to eliminate unnecessary ‘for’ loop iterations and complexity. Also, it is written in lower level languages like C, it is faster than conventional Python ‘for’ loops. Since I am using minimum number of variable to store data during the program running, the space efficiency also in good standing.

Question 2: Explanation

In this function, the longest palindromic substring is found in a given string. To achieve this, a substring is divided in to two from the center and check whether they are backwardly matching or not.

The overall efficiency of this function also depend on the length of the string.