Word Ladder I

Given two distinct words **startWord** and **targetWord**, and a list denoting **wordList** of unique words of equal lengths. Find the length of the shortest transformation sequence from startWord to targetWord.
Keep the following conditions in mind:

- A word can only consist of lowercase characters.
- Only one letter can be changed in each transformation.
- Each transformed word must exist in the wordList including the targetWord.
- startWord may or may not be part of the wordList

The second part of this problem can be found here.

Note: If no possible way to transform sequence from startWord to targetWord **return 0**

Example 1:

```
Input:
wordList = {"des","der","dfr","dgt","dfs"}
startWord = "der", targetWord= "dfs",
Output:
3
Explanation:
The length of the smallest transformation
sequence from "der" to "dfs" is 3
i,e "der" -> "dfr" -> "dfs".
```

Example 2:

```
Input:
```

```
wordList = {"geek", "gefk"}
startWord = "gedk", targetWord= "geek",
Output:
2
Explanation:
gedk -> geek
```

Example 3:

```
Input:
wordList = {"poon", "plee", "same",
"poie","plea","plie","poin"}
startWord = "toon", targetWord= "plea",
Output: 7
Explanation:
toon -> poon -> poin -> poie -> plie -> plee -> plea
```

Your Task:
You don't need to read or print anything, Your task is to complete the function **wordLadderLength()** which takes startWord, targetWord and wordList as input parameter and returns the length of the shortest transformation sequence from startWord to targetWord. If not possible return 0.

Expected Time Compelxity: O(N^2 * M)
Expected Auxiliary Space: O(N * M) where N = length of wordList and M = |wordList_i|

Constraints:
$1 \leq N \leq 100$
$1 \leq M \leq 10$