1. Write a function called `read_dictionary()` that opens dictionary.txt, reads in all the words into a list, and returns the list.

2. Write a function called `count_no_match` that takes two string parameters. If the two strings have different lengths, return -1. If the two strings have the same length, then compare the strings letter by letter and count the number of letter pairs between the strings that don't match. Return this number.

   Example: `count_no_match("pen", "pong")` returns -1 because the strings have different lengths.
   Example: `count_no_match("pen", "pen")` returns 0 because the strings are the same.
   Example: `count_no_match("pen", "pem")` returns 1 because p matches p, e matches e, but n and m don't match.
   Example: `count_no_match("the", "teh")` returns 2 because t matches t, but h doesn't match e, and then e doesn't match h.

   ```
   def count_no_match(str1, str2):
   ```

3. Write a function called `get_matches(dict_list, word)` that FILTERS dict_list to find all words in the dictionary that have exactly one letter different from word. (dict_list will be the list of words from the dictionary.)

   Example: `get_matches(dict_list, "houled")` should return
   `['fouled', 'hauled', 'housed', 'howled', 'souled']`

   ```
   def get_matches(dict_list, word):
   ```

4. Write a function called `autocorrect` that takes a dictionary list, and a text message as a list of words and returns an autocorrected text message as a list. This is a LIST TRANSFORM operation. For each word in the text message, if the word is in the dictionary, leave it alone. If it's not in the dictionary, get the list of best matches for the word and pick one *at random* to change it to. If the list of best matches comes back empty for a word, then leave the word alone.

   Ex: `autocorrect(dict_list, ["i", "loke", "computars"])` might return `["i", "like", "computers"]`

   ```
   def autocorrect(dict_list, message):
   ```

5. Write a main function that lets the user type in a text message and it will be autocorrected.

6. Challenge: This version of autocorrect breaks if it can't find a word that differs from a misspelled word in only one letter. For instance, "scienec" cannot be autocorrected to "science" because they differ in TWO letter positions. Change your get_matches function so that if there are no matches that differ in one letter, it tries two letters, then three letters, etc.

7. Challenge: This version of autocorrect doesn't handle the common case of "swapped" letters very well. For instance, "teh" is a common misspelling of "the," but our autocorrect will try to change "teh" to "tea" or "ten" or another word that differs in only one letter. Change your get_matches function (you may want to write a separate function) to not penalize letter swaps twice the way count_no_match does.