Algorithms & Data Structures

LAB 6
LEVENSHTEIN DISTANCE
(3p + 1p + 1p (challenge))

1. The file ocr_output.txt contains the result of the work of some (fictious) OCR engine. As it can be easily noticed, this file contains lots of erroneous words. For each word *w* not found in the given dictionary (wrt.dic), present a list of dictionary suggestions, i.e. the words most similar to the current word according to the Levenshtein distance. If for example this distance is 1, then list all the dictionary entries with Levenshtein distance to *w* equal 1.

   Note that the input text contains only lowercase words and the punctuation marks (only '.' and ',') are prepended with a space (for parsing simplicity). Do not try to find suggestions for "words" (including those punctuation marks) of length 1.

2. Show the statistics of your program running on ocr_output.txt with wrt.dic dictionary:
   - what % of the words from ocr_output.txt were unknown,
   - the average suggestion list length,
   - the average Levenshtein calculation time.

Challenge.
Order the suggestions in step 1, according to the common prefix length.
Example:
the input word: *wanter*
suggestions returned in step 1: *waiter*, *wander* (both with Lev distance == 1),
but as the longer common prefix belongs to *wander* ("wan"), the suggestions are shown in the order: *wander*, *waiter*.

For sorting suggestions, use C library function qsort or C++ STL sort (DON'T use your own sort routine).

(words.dic is is part of WRT-ENG.dic dictionary from WRT 4.6 compression software, by Przemysław Skibiński, http://www.ii.uni.wroc.pl/~inikep/research/WRT/WRT46.zip)