Algorithms & Data Structures

LAB 4
HASHING (with open addressing)
(2p + 2p (+ 1p + 1p challenge))

1. Given the file words20k.txt (one word per line, all words unique, 20,000 words in the file),
   a) read the words one by one into a string array A;
   b) create another array (H) of size 24,000, fill with empty strings (i.e. "") first, and then copy items from A into H, using a hash function and linear probing for collision resolution.
      The hash function to use, for string s:
      $h'(s) = 39 * int(s[0])$, if s.size() == 1,
      $h'(s) = 39 * int(s[0]) + 39^2 * int(s[1])$, if s.size() == 2,
      $h'(s) = 39 * int(s[0]) + 39^2 * int(s[1]) + 39^3 * int(s[2])$, if s.size() == 3,
      $h'(s) = 39 * int(s[0]) + 39^2 * int(s[1]) + 39^3 * int(s[2]) + 39^4 * int(s[3])$, otherwise,
      and then
      $h(s) = h'(s) \% 24000$;
   c) measure the average item insertion time for the first 500 inserted words, for the next 500 inserted words, etc. until the last 500 inserted words.

   NOTE. For time measurements (under Windows) use the precise method from http://szgrabowski.kis.p.lodz.pl/Alg13/precise.cpp

2. In a loop, read 1000 words from array A (say, from index 14000 up to 14999), search for them in the hash array H and delete (one by one). Present the avg delete time, and the number of collisions: max, min and avg over those 1000 searches (until the given element is found and then deleted). Then ask the user for entering a word *w* and look for it in H, presenting all the collisions that have happened before finding *w* (if *w* exists in H at all!).

Challenge

3. Replace the inefficient hash function from 1b) with djb2 and then sdbm functions from http://www.cse.yorku.ca/~oz/hash.html (adapt the presented code to your data types). How does it affect the average number of collisions? Try to explain the results.

4. Repeat the experiment from 1–2, but with using double hashing instead of linear probing. Propose your own functions for double hashing. Plot the results (first experiment vs. this one).

(words20.txt 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)