CS 211 Data Structures and Algorithms Lab
July -- December, 2018
Assignment 6
Total Marks: 10
Due on 6th October

An anagram is a word formed from another by rearranging its letters. For example 'brainy' is an anagram of 'binary' and vice versa. The objective of this assignment is to find anagrams using hashing.

Command-line argument:
Your program should receive three command-line arguments: a file containing words, the size of hash table, and a query file. For example, a typical run of your program can be ./a.out words.txt 25000 query.txt .

Input

A file containing words: The first command-line argument is the path of a text file containing English words. The file contains one word per line.
The size of hash table: The second command-line argument is a positive integer which denotes how many slots should be there in the hash table you create.
A file containing queries: The third command-line argument is a text file containing a list of 'words', one per line. Here a word may not have any meaning.

Task

Let M be the size of the hash table (given by the second command-line argument). Use the hash function h described below. Given a string, the hash function sums the ASCII values of the characters in the string and take the remainder obtained by dividing the sum with M. For example h('brainy') = 98+114+97+105+110+121 % M. If M=25000, this value is 645.

Your hash table should resolve collisions by chaining. Every slot in the hash table should contain a pointer to a linked list. Take the words one by one from the input file containing words, apply the hash function, and insert the word in the linked list associated with the slot given by the hash function. The insertion should always be done at the beginning of the linked list.

Take every 'word' in the file containing queries, hash it, and list all anagrams of the 'word' present in the linked list associated with the slot given by the hash function. Note that the hash function we defined will hash all anagrams of a word to the same slot. So, to obtain the anagrams of a word, it is enough to search in the linked list associated with the slot. Also note that there can be other words, which are not anagrams in the same linked list. Your program should able to ignore them while collecting all anagrams of the word.

Output

The output of your program should be in a file named 'anagrams.txt' Corresponding to every line in the input query file, there should be a line in the output file. As mentioned before, every line in the query file contains a 'word', which may or may not be meaningless and may or may not be in the file containing the words. The output line for a word is the list of all anagrams (of the word) present in the file containing words. Note that, if the query word is in the words file, then the word is also listed as an anagram of the word and if the query word is not in the words file, then it is not listed as an anagram of the word. The anagrams of a word are listed in a single line where two anagrams of the same word are separated by a single white space. The anagrams are listed in the reverse order of the order given in the words file. That means if the query word is 'brainy', the output line contains 'brainy' before 'binary' (assuming 'brainy' comes after 'binary' in the words file). Note that this requirement is natural as you are asked to insert always at the beginning of the linked list associated with a slot.

Submission and Evaluation

- The program you submit should output a file named 'anagrams.txt' when run.
- There should be only one main file and it should be named as main.<extension>, where the extension depends on the language you choose (You must use either C or C++).
- Test well before submission. We have some hidden inputs with us to test your program. The mark you obtain is purely based on whether your program correctly gives outputs for the hidden inputs.
- Submit your code as a zip file (even if there is only one file) where the name of the zip file is your roll number. It is important that you follow the input/output conventions exactly (including the naming scheme) as we may be doing an automated evaluation.
- This assignment is due on 6th October. Penalty for late submission is 5% per week; i.e., if you submit on 8th October, you will get only 95% of the mark you deserve otherwise.
- Follow some coding style uniformly. Provide proper comments in your code.
- Submit only through Moodle. Submit well in advance. Any hiccups in the Moodle/internet at the last minute is never acceptable as an excuse for late submission.
- Acknowledge the people (other than the instructor and TAs) who helped you to solve this assignment. The details of the help you received and the names of the people who helped you (including internet sources, if applicable) should come in the main file or in a separate file (acknowledge.txt). Copying others' programs is a serious offence.
- Honesty policy of the institute will be strictly followed. Note that we have access to a very good software to check plagiarism.