MAJOR ASSIGNMENT 2

COMPARE SORTED LINKED LISTS WITH HASH TABLES

Name: Jainish Prakash Patel

Date of Submission: 11-07-2022

Data Structures

SENG1050

Checklist

Aim: Compare the efficiency of searching a sorted linked list and a hash table.

1. Create and use linked lists.
2. Create and use hash tables.

Requirements:

1. Changed Requirements:
    - Instead of getting words to load into the hash table from the user, you must load at least 2000 names from a file called names.txt as input (using file I/O). Names will have spaces within the strings. It is OK if you share the names with classmates for this.
    - Make sure that your names are in random order (you can use Excel for this). The names.txt file must be in the same directory as your source code. All names should be in lowercase.
    - It's OK to use more than 2000 names but don't have more than 10000.
    - The hash table's buckets must point to sorted linked lists. When you load a name into the hash table, also load the names into one very long sorted linked list.
    - You must call the same function for inserting into the sorted linked list as for inserting into the hash table bucket's linked list. Do not duplicate code for linked list insertion.
    - Make the hash table 127 buckets in size.
    - You must create a function called searchLinkedList() that searches a sorted linked list for a name and returns a pointer to the node containing the name (if found) or NULL.
    - This function must return NULL immediately after you pass the point in the linked list where the name would have been found if it was in the linked list.
    - The function takes three parameters:
    - char * searchName (or you can use a string object): name to search for (entered by the user)
    - struct linkedList * linkedList: linked list to search (in your program, you can call the linked list node struct anything that makes sense)
    - int * comparisonCount: pointer to int filled in by the function with the count of strcmp comparisons done in searching the linked list

- Again, you can assume that all names will be in lowercase.
- Unlike Focused Assignment 3, do not display the words that you compare against in the search.
- Create a search function called searchForNameTwice(). It returns nothing and will have the following parameters:
- char * searchName (or you can use a string object): name to search for (entered by the user)
- struct linkedList * linkedList: linked list to search
- struct linkedList * hashTable[]: hash table to search
- int comparisonCount[2]: array containing the count of strcmp comparisons done in searching the extremely-long sorted linked list (element 0) and in searching the hash table (element 1)
- It must call your linked list search function. It then displays one of the following messages once the search is done:
- "name was found in the list in number comparisons", where name is the name being searched for, list is either "linked list" or "hash table bucket" and number equals the number of times that strcmp was called "name was NOT found in the list in number comparisons"
- You will use this search function to search the hash table bucket and the sorted linked list.
- Don't worry about the grammatical inconsistency of possibly displaying "1 comparisons".
- Indent the message displayed by one TAB ('\t').
- Once you are finished the loop, display three blank lines. Then display the total number of searches, the total number of comparisons done by searching the sorted linked list, and the total number of comparisons done by searching the hash table on three separate lines.
- Clean up all allocated memory before exiting .
- Create a JPG file called compare.jpg that contains a screenshot of your program running with a full screen of sample output.