SENG 265: Programming Exercise 1

DUE: Monday 6 June in lab

Provided Files

- word_storage_v1.2.zip - Word Storage directory tree (UPDATED)
- performance_chart.ods - Performance chart template

Problem Summary

Your task is to produce a new version of the Word Storage Module by:

1. modifying the implementation to use hashing, rather than binary search and
2. carrying out a performance evaluation comparing the execution times of `ws_find_word` in the current Word Storage Module and in your new implementation.

This is a team assignment; it must be completed by the teams formed in lab.

Downloading the Word Storage Module

1. Use your browser to download the Word Storage directory tree.
2. From the directory where you have placed the zip file, unpack and build all the executables by typing:

   ```
   unzip word_storage_v1.2.zip
   cd word_storage_v1.2
   make
   ```

3. Run the Word Storage Test Implementation by typing:

   ```
   ./word_storage_test
   ```

   There should be no output.
4. Carefully study all of the source files in the `word_storage_v1.2` directory.

Modify the Word Storage Module Implementation

In the current implementation of the Word Storage Module, the data structure is an array of strings, stored in ascending order. The array is searched using binary search

Your task is to replace the array of strings and binary search with hashing, as described next:

1. The current array of strings must be replaced by an array of list headers. The following definitions must be used:

   ```
   struct word_node {
           char *word;
           struct word_node *next;
   };
   #define HASH_TABLE_SIZE 1000
   static struct word_node *word_hash_table[HASH_TABLE_SIZE];
   ```

2. The hash function must have the following prototype:
   `static int hash(char *word)`
   When called, `hash(S)`, must return the following value: the sum of the ASCII values in string *S* modulo `HASH_TABLE_SIZE`. When computing this sum, alphabetic characterers must be "normalized" by converting each upper case letter to lower case.
3. Each time a new word *W* is added, it must be placed in the linked list headed by `word_hash_table[hash(W)]`. For example, if HASH_TABLE_SIZE is 100, then `hash("aB"))` must return (97+98)%100 = 95.
4. When a search is conducted for word *W*, the search is done only on the elements in the linked list headed by `word_hash_table[hash(W)]`.

Carry Out a Performance Evaluation: find

Your task is to compare the execution time of the new and old versions of the `ws_find_word` function: The code in `word_storage_find_test.c` measures the time required to find a word. Typing:

```
./word_storage_find_test N
```

adds N words. Then `ws_find_word` is called, with a word larger than any of the N words previously finded to the module. The time required for the last `ws_find_word` call is written to standard output. You must conduct 11 tests, with N = 0, 10000, 20000, ..., 100000. Present your results in an OpenOffice chart contained in the file `performance_chart.ods`.

Display the number of words, N, on the X axis. On the Y axis, display the execution time, in microseconds, for the last `ws_find_word` call. Provide two plotted lines: one for the old Word Storage implementation and one for your new version.

Create the chart using Calc, the OpenOffice spreadsheet application, to modify the table in file `performance_chart.ods`.

Grading Criteria

Your mark will be based on the following criteria:

[60%] Code correctness
- Is your new module implementation correct with respect to the Word Storage Module Specification?
- Are there memory leaks in your code?

[15%] Code and documentation style
- Is your code as simple as possible? In particular, your version of `word_storage.c` should not exceed 200 lines in length, as measured by the `wc` command.
- Have you updated the code comments to be consistent with the code changes?
- Have you used code and documentation style consistent with the existing code?

[15%] Performance measurement
- Have you modified `performance_chart.ods` as directed? Are the performance results reasonable?

[10%] Assignment submission
- Have you followed the instructions for submitting the assignment?

Submitting your Solution

You must submit a single zip file

```
N.word_storage.zip
```

where N is the login you use in the SEng 265 lab.

The zip file must contain the following files and no others:

- `word_storage.c`
- `performance_chart.ods`