Jack McVeigh

CS 260

22 October 2021

Programming Assignment 2: Spell Check Report

Introduction

Spell check is a well-known feature in all forms of textual user input, whether it be in texting, emailing, document editing, or programming via an IDE, spell check provides a helpful oversight when inputting thoughts into a keyboard. Spell check can be implemented in many ways via a hash table, and depending on the algorithms used in the process, spell checkers can provide near perfect recommendations to a user. The chosen implementation for a spell check program can be observed in the below flow chat, Figure 1.

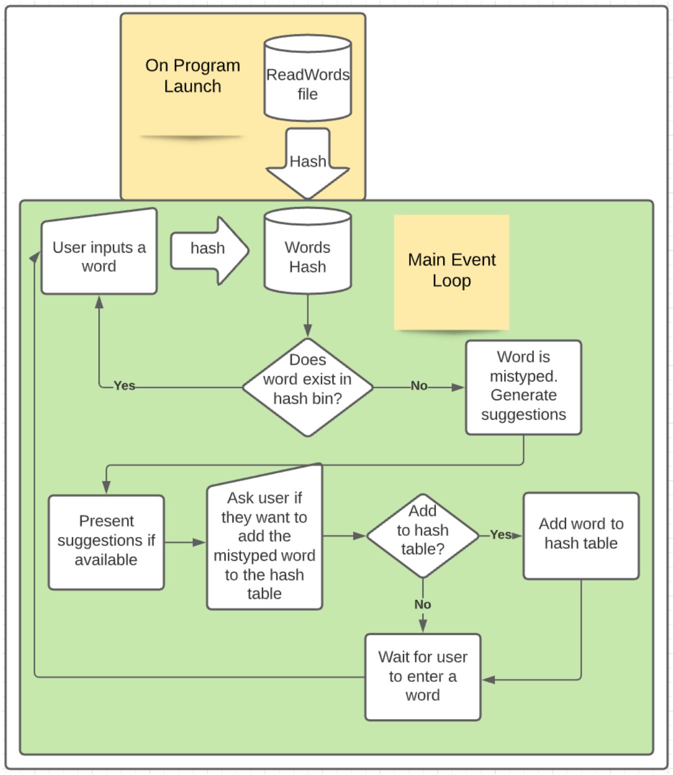
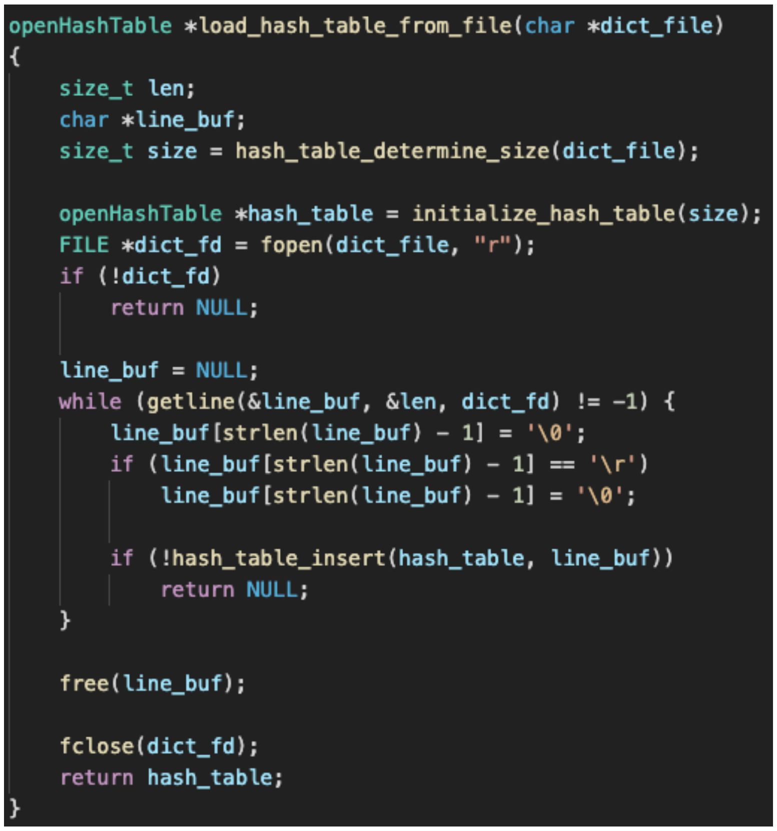

On Program Launch
ReadWords file
Hash
User inputs a word
hash
Words Hash
Main Event Loop
Does word exist in hash bin?
Yes
No
Word is mistyped. Generate suggestions
Present suggestions if available
Ask user if they want to add the mistyped word to the hash table
Add to hash table?
Yes
Add word to hash table
No
Wait for user to enter a word

Figure 1. Spell Check program execution.

Loading the Hash Table

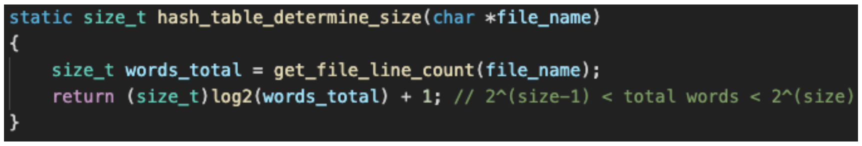
The hash table is loaded from a file by reading the file line by line, hashing each word and pushing the word to the correct linked list based on the hash value. Figure 2 contains the function that runs this process.



```c
openHashTable *load_hash_table_from_file(char *dict_file)
{
    size_t len;
    char *line_buf;
    size_t size = hash_table_determine_size(dict_file);

    openHashTable *hash_table = initialize_hash_table(size);
    FILE *dict_fd = fopen(dict_file, "r");
    if (!dict_fd)
        return NULL;

    line_buf = NULL;
    while (getline(&line_buf, &len, dict_fd) != -1) {
        line_buf[strlen(line_buf) - 1] = '\0';
        if (line_buf[strlen(line_buf) - 1] == '\r')
            line_buf[strlen(line_buf) - 1] = '\0';

        if (!hash_table_insert(hash_table, line_buf))
            return NULL;
    }

    free(line_buf);

    fclose(dict_fd);
    return hash_table;
}
```

Figure 2. Loading an open hash table via a file.

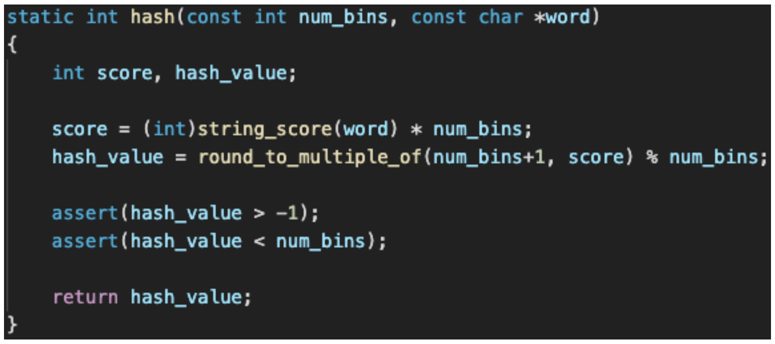
The size of the hash table is dynamically determined via the process in Figure 3. The hash table's size grows logarithmically to provide higher suggestion hit rates as the size of the input dictionary grows.



```c
static size_t hash_table_determine_size(char *file_name)
{
    size_t words_total = get_file_line_count(file_name);
    return (size_t)log2(words_total) + 1; // 2^(size-1) < total words < 2^(size)
}
```

Figure 3. Determine the size of the hash table.

Figure 4 contains the hashing algorithm selected for this implementation.

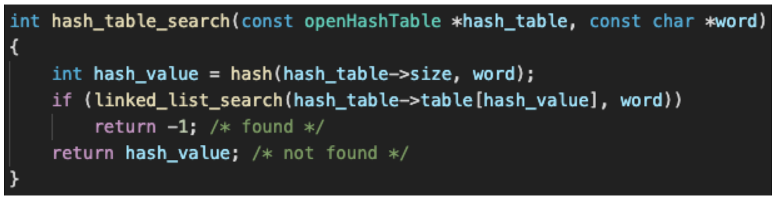


```c
static int hash(const int num_bins, const char *word)
{
    int score, hash_value;

    score = (int)string_score(word) * num_bins;
    hash_value = round_to_multiple_of(num_bins+1, score) % num_bins;

    assert(hash_value > -1);
    assert(hash_value < num_bins);

    return hash_value;
}
```

Figure 4. The hashing function. String score is the sum of each of a word's character's ascii value divided by its length.

Determine if Input is in the Hash Table

Searching the hash table for the given word is as easy as just hashing the word and searching the corresponding linked list for the word. If the word is in the linked list, the word is a correctly, typed word. If not, the word is mistyped.

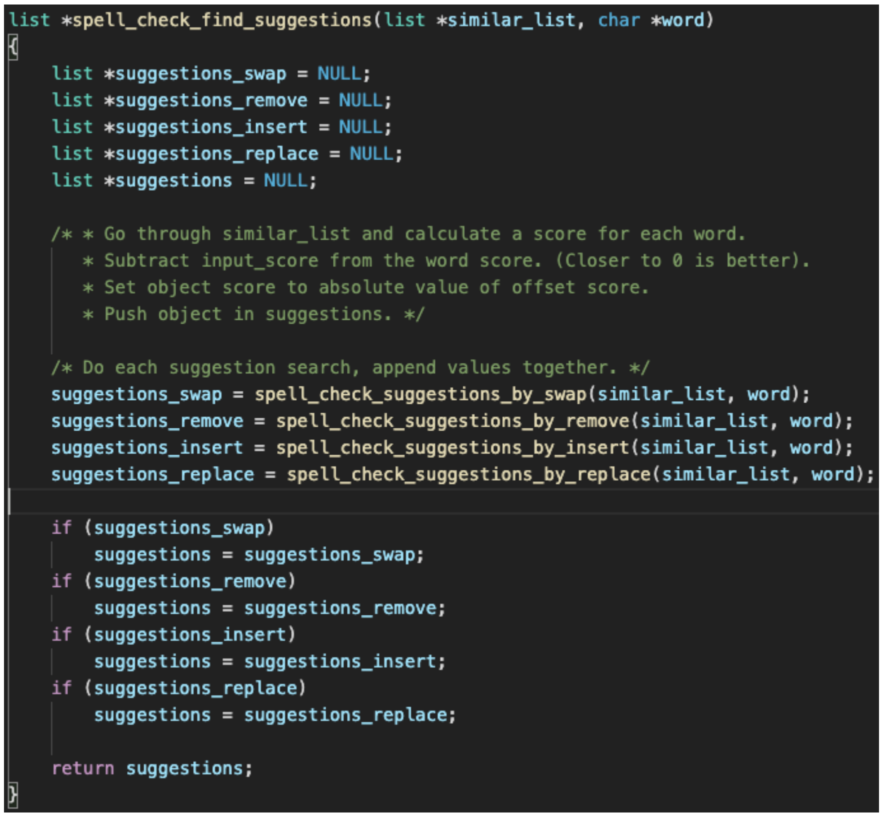


```c
int hash_table_search(const openHashTable *hash_table, const char *word)
{
    int hash_value = hash(hash_table->size, word);
    if (linked_list_search(hash_table->table[hash_value], word))
        return -1; /* found */
    return hash_value; /* not found */
}
```

Figure 5. Searching the hash table for a given word.

Determine Suggestions for Misspelled Word

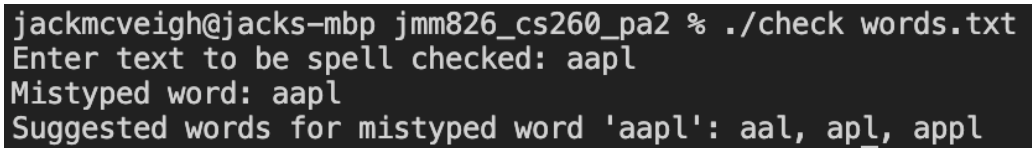
Finding suggestions for the user is done by taking the hash value of the input word and manipulating the corresponding linked list's words to match the input word. If a word in the dictionary can be manipulated by the means of swapping letters, removing letters, inserting extra letters, or replacing existing letters, then the candidate word can be provided to the user as a suggestion. Figure 6 shows this process.



```
list *spell_check_find_suggestions(list *similar_list, char *word)
{
    list *suggestions_swap = NULL;
    list *suggestions_remove = NULL;
    list *suggestions_insert = NULL;
    list *suggestions_replace = NULL;
    list *suggestions = NULL;

    /* * Go through similar_list and calculate a score for each word.
       * Subtract input_score from the word score. (Closer to 0 is better).
       * Set object score to absolute value of offset score.
       * Push object in suggestions. */

    /* Do each suggestion search, append values together. */
    suggestions_swap = spell_check_suggestions_by_swap(similar_list, word);
    suggestions_remove = spell_check_suggestions_by_remove(similar_list, word);
    suggestions_insert = spell_check_suggestions_by_insert(similar_list, word);
    suggestions_replace = spell_check_suggestions_by_replace(similar_list, word);

    if (suggestions_swap)
        suggestions = suggestions_swap;
    if (suggestions_remove)
        suggestions = suggestions_remove;
    if (suggestions_insert)
        suggestions = suggestions_insert;
    if (suggestions_replace)
        suggestions = suggestions_replace;

    return suggestions;
}
```

Figure 6. Manipulate word in corresponding linked list to find suggestions.

After the suggested words are determined and if any are found, they are presented to the user. Figure 7 shows the output of the whole process.

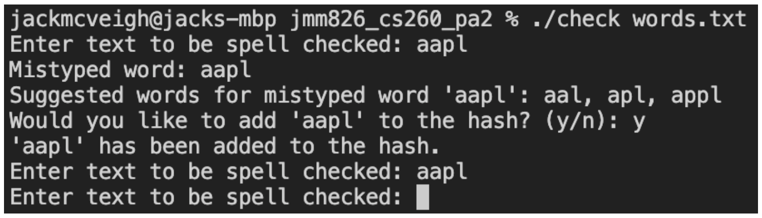


```
jackmcveigh@jacks-mbp jmm826_cs260_pa2 % ./check words.txt
Enter text to be spell checked: aapl
Mistyped word: aapl
Suggested words for mistyped word 'aapl': aal, apl, appl
```

Figure 7. The output of the process of finding suggestions.

Ask User to Write Mistyped word Back to the Hash Table

Before continuing the program to prompt the user for another word, first the program asks the user if they would like to include their input word in the dictionary.



```
jackmcveigh@jacks-mbp jmm826_cs260_pa2 % ./check words.txt
Enter text to be spell checked: aapl
Mistyped word: aapl
Suggested words for mistyped word 'aapl': aal, apl, appl
Would you like to add 'aapl' to the hash? (y/n): y
'aapl' has been added to the hash.
Enter text to be spell checked: aapl
Enter text to be spell checked: 
```

Figure 8. Writing input word back to the hash table.

If the user opts to write the word back to the hash table, subsequent runs will be able to check against this newly added word.

Conclusion

The running time of the spell check algorithm implemented in this project for correctly spelled words is 0(n), where n is the length of the corresponding linked list for the hashed input word. The running time of the algorithm for mistyped words is O(n^2) as many more operations must be done to determine the suggestion words. To better implement this algorithm and to obtain better results with the same time complexity, a better scoring system for suggestion words must be developed. This implementation lacks that capability due to time constraints.