Project Report

Spring 2025

Name: Rakshith
Project: Spelling

Note: As you write each section, try to be as clear and detailed as possible. Your goal is to communicate your thought process and work clearly. Don't worry if you faced challenges or made mistakes; discussing these is a valuable part of learning and shows your problem-solving skills! Remember, there's no single 'right' way to do these tasks, so be creative and honest in your responses.

Problem Statement (2-3 Paragraphs):

The goal of this project was to build a spelling checker that reads a text file word-by-word and compares every word against a dictionary file. The main purpose was to identify all of the words in the text that do not appear in the dictionary, then report them to the user and save them in an output file. The program also needed to keep track of every correctly spelled word and count how many times each one appears.

The inputs for the program are the name of the text file the user wants to check and the dictionary file that contains all valid English words. The output includes a list of all the misspelled words printed on the screen, along with two output files: one listing every misspelled word and another listing all correctly spelled words with their counts. Because the assignment does not guarantee perfect input, I considered that the file might contain punctuation, uppercase letters, or blank tokens. To handle this, the code strips punctuation from the beginning and end of each word and converts everything to lowercase before comparison.

The main problem was trying to open files from the wrong directory. I made sure to run the program inside the same folder as the input files to avoid issues related to finding the right files. I also added checks to skip empty strings so that the program would not crash or miscount data.

Design (1-3 Paragraphs):

I divided the program into four main functions to keep it organized and easy to follow. The load_words function loads the dictionary into memory. The clean_token function strips punctuation from the start and end of each token. The check_spelling function handles the main logic of reading each word, checking it against the dictionary, and writing the results. Finally, main only handles user input and function calls. This kept the program simple and aligned with what we’ve learned through Chapter 7.

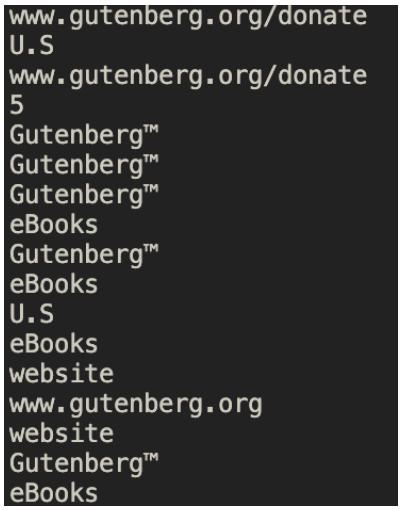
The design is straightforward because it uses basic lists, loops, and string functions. Storing the dictionary words in a list and checking each word with a lowercase comparison allowed the program to run clearly and without confusion. Counting words was handled with a dictionary that maps each correctly spelled word to its frequency. Sorting the correct words alphabetically before writing them to a file made the output easier to read.

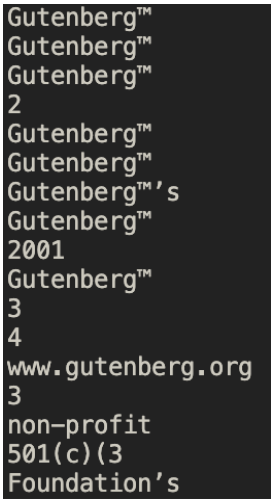
Looking back, the approach worked well. If I were to improve it in the future, I might expand the punctuation stripping to handle more unusual characters or let the user choose output filenames. Overall, the current structure is effective for this level of the course.

Testing (1-2 Paragraphs + screenshots of 3 test cases):

To test the program, I started by running it with the sample file book.txt, since that file includes a wide variety of words, punctuation, and formatting. This let me confirm that the program could handle long text, special characters, and repeated words.

Normal inputs included typical sentences containing basic English words that I knew were in the dictionary. Special cases included words with punctuation attached, uppercase words, numbers, URLs, and lines with blank spaces. The program successfully stripped punctuation, converted words to lowercase, and correctly identified misspelled items. The output files were created as expected, and both the printed output and the file content matched the assignment's sample test case.





Conclusion (1 paragraph)

This project worked as intended and produced correct results for every test I ran. I learned how to combine string processing, file input/output, and data structures like dictionaries into one program. The project was successful because it followed the assignment instructions exactly, produced the required output files, and matched the sample test cases. In future projects, I would continue using separate functions to keep my code organized, and I would consider

adding more error handling or more flexible punctuation cleaning to improve the program even further.