**CS-221 LAB Data Structure & Algorithm**

**Semester Project Report**

**Mr. Usama Arshad**

**Ghulam Ishaq Khan Institute of Engineering Sciences & Technology, Topi, Swabi 23460, Pakistan**

**Spell Checker in C++**

**Members:**

* **2021438** (Muhammad Rehan)
* **2021051** (Adeen Amir)
* **2021355** (Muhammad Arsal)

**Brief:**

Implement a spell checker using a hash table and a trie. The hash table can be used to store the dictionary of words, while the trie can be used to efficiently search for words and suggest corrections for misspelled words.

**Overview:**

High-level overview of how you could implement a spell checker using a hash table and a trie in C++:

First, you will need to create a hash table to store the dictionary of words. You can use a library like unordered set from the C++ Standard Template Library (STL) to implement the hash table.

Next, you will need to create a trie data structure to store the words in the dictionary. A trie is a tree-like data structure that stores a set of strings, such that the path from the root to any node represents a prefix of a string stored in the trie.

Once you have implemented the hash table and the trie, you can begin building the spell checker. To do this, you will need to read in the dictionary of words and insert each word into both the hash table and the trie.

When the user inputs a word to be checked, you can first check if it is in the hash table. If it is, then it is spelled correctly, and you can return this result to the user. If it is not in the hash table, you can use the trie to search for similar words and suggest corrections to the user.

To search for similar words in the trie, you can traverse the trie and compare the prefixes and suffixes of the words in the trie to the input word. If the prefixes match, you can add the word to a list of potential corrections. You can then present this list to the user and let them choose the correct word,

or you can use some heuristics (e.g., word frequency, edit distance) to automatically select the most likely correction.

**Summary:**

This is just a high-level overview of how you could implement a spell checker using a hash table and a trie in C++. There are many details and considerations that you will need to consider when implementing this project, such as how to oversee diverse types of errors (e.g., transpositions, deletions, insertions) and how to efficiently implement the trie data structure.