**Assignment 3**

**Code:**

#include <iostream>

#include <fstream>

#include <sstream>

#include <map>

#include <vector>

#include <string>

#include <algorithm>

using namespace std;

// Function to clean up a word by removing punctuation and converting to lowercase

string cleanWord(const string &word) {

    string cleaned = word;

    cleaned.erase(remove\_if(cleaned.begin(), cleaned.end(), ::ispunct), cleaned.end());

    transform(cleaned.begin(), cleaned.end(), cleaned.begin(), ::tolower);

    return cleaned;

}

// Simple implementation of Porter Stemmer

string stemWord(const string &word) {

    string stemmed = word;

    // This is a simplified version; real implementations are more complex.

    // Remove common suffixes

    if (stemmed.size() > 4) { // Only stem words with length > 4 to avoid excessive shortening

        if (stemmed.substr(stemmed.size() - 3) == "ing")

            stemmed = stemmed.substr(0, stemmed.size() - 3);

        else if (stemmed.substr(stemmed.size() - 2) == "ed")

            stemmed = stemmed.substr(0, stemmed.size() - 2);

        else if (stemmed.substr(stemmed.size() - 1) == "s")

            stemmed = stemmed.substr(0, stemmed.size() - 1);

    }

    return stemmed;

}

class InvertedIndex {

    map<string, vector<int>> index;  // The inverted index

public:

    // Function to build the index from a file

    void buildIndex(const string &filename) {

        ifstream file(filename);

        string line;

        int lineNumber = 0;

        while (getline(file, line)) {

            lineNumber++;

            stringstream ss(line);

            string word;

            while (ss >> word) {

                word = cleanWord(word);     // Clean the word

                word = stemWord(word);      // Stem the word

                index[word].push\_back(lineNumber); // Add line number to the word's occurrence list

            }

        }

        file.close();

    }

    // Function to search for a word in the index

    void search(const string &query) {

        string cleanedQuery = cleanWord(query);

        string stemmedQuery = stemWord(cleanedQuery);

        if (index.find(stemmedQuery) != index.end()) {

            cout << "The word '" << query << "' (stemmed to '" << stemmedQuery << "') is found at lines: ";

            for (int lineNum : index[stemmedQuery]) {

                cout << lineNum << " ";

            }

            cout << endl;

        } else {

            cout << "The word '" << query << "' was not found in the document." << endl;

        }

    }

    void printIndex() {

        cout << "Inverted Index:\n";

        for (const auto &entry : index) {

            cout << entry.first << ": ";

            for (int lineNum : entry.second) {

                cout << lineNum << " ";

            }

            cout << endl;

        }

    }

};

int main() {

    InvertedIndex index;

    string filename = "3a.txt";  // Name of the text file

    // Build the inverted index

    index.buildIndex(filename);

    index.printIndex();

    // Perform searches

    string query;

    cout << "Enter a word to search: ";

    cin >> query;

    index.search(query);

    return 0;

}

**OUTPUT:**

Inverted Index:

a: 3 3

agility: 4

and: 2 5

are: 4

away: 5

brown: 1 3 5

dog: 1

for: 4

fox: 1 2 3 5

foxe: 4

is: 2 3

jump: 1 5

known: 4

lazy: 1

over: 1

quick: 1 2 3 5

rare: 3

sight: 3

smart: 2

the: 1 1 2 5

their: 4

was: 5

Enter a word to search: brown

The word 'brown' (stemmed to 'brown') is found at lines: 1 3 5