**Assignment 1**

**Code**

#include <iostream>

#include <fstream>

#include <sstream>

#include <unordered\_set>

#include <vector>

#include <unordered\_map>

#include <algorithm>

#include <cctype>

using namespace std;

// Function to check if a word is a stop word

bool isStopWord(const string &word, const unordered\_set<string> &stopWords) {

    return stopWords.find(word) != stopWords.end();

}

// Simple stemming function to remove common suffixes

string stemWord(const string &word) {

    string stemmedWord = word;

    if (stemmedWord.length() > 4) {

        if (stemmedWord.substr(stemmedWord.length() - 3) == "ing") {

            stemmedWord = stemmedWord.substr(0, stemmedWord.length() - 3);

        } else if (stemmedWord.substr(stemmedWord.length() - 3) == "ion") {

            stemmedWord = stemmedWord.substr(0, stemmedWord.length() - 3);

        } else if (stemmedWord.substr(stemmedWord.length() - 2) == "ed") {

            stemmedWord = stemmedWord.substr(0, stemmedWord.length() - 2);

        }

    }

    return stemmedWord;

}

int main() {

    // Read stop words from file

    unordered\_set<string> stopWords;

    ifstream stopFile("stopwords.txt");

    string stopWord;

    while (stopFile >> stopWord) {

        stopWords.insert(stopWord);

    }

    stopFile.close();

    // Read the document from file

    ifstream docFile("document.txt");

    stringstream buffer;

    buffer << docFile.rdbuf();

    string document = buffer.str();

    docFile.close();

    // Remove punctuation from document

    for (char &c : document) {

        if (ispunct(c)) c = ' ';

    }

    // Process the document to remove stop words and apply stemming

    stringstream ss(document);

    string word;

    unordered\_map<string, int> wordFrequency;

    vector<string> processedWords; // Store processed words for printing the sentence

    while (ss >> word) {

        // Convert to lowercase

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

        // Check if the word is a stop word

        if (!isStopWord(word, stopWords)) {

            // Apply stemming

            string stemmedWord = stemWord(word);

            // Store the processed word for printing the sentence

            processedWords.push\_back(stemmedWord);

            // Count word frequency

            wordFrequency[stemmedWord]++;

        }

    }

    // Print the processed output sentence

    cout << "Processed sentence: ";

    for (const string &w : processedWords) {

        cout << w << " ";

    }

    cout << endl;

    // Print the word frequencies

    cout << "\nWord frequencies after stop word removal and stemming:" << endl;

    for (const auto &pair : wordFrequency) {

        cout << pair.first << ": " << pair.second << endl;

    }

    return 0;

}

**OUTPUT**

**Processed sentence: quick brown fox jump over lazy dog dog very lazy tir foxes like jump play woods**

**Word frequencies after stop word removal and stemming:**

**play: 1**

**like: 1**

**foxes: 1**

**very: 1**

**woods: 1**

**lazy: 2**

**tir: 1**

**dog: 2**

**fox: 1**

**over: 1**

**jump: 2**

**brown: 1**

**quick: 1**