Assignment No:3

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <vector>
#include <map>
#include <set>
#include <algorithm>

using namespace std;

vector<string> readDocuments(const string &filename) {
        vector<string> documents;
        ifstream file(filename);

        if (!file.is_open()) {
        cerr << "Error: Could not open file " << filename << endl;
        return documents;
        }

        string line;
        while (getline(file, line)) {
        documents.push_back(line);
        }

        file.close();
        return documents;
}

void printDocuments(const vector<string> &documents) {
        cout << "Documents:\n";
        for (size_t i = 0; i < documents.size(); ++i) {
        cout << "Document " << i << ": " << documents[i] << "\n";
        }
}

map<string, set<int>> buildInvertedIndex(const vector<string> &documents) {
        map<string, set<int>> invertedIndex;

        for (size_t docId = 0; docId < documents.size(); ++docId) {
        istringstream iss(documents[docId]);
        string word;
```

```
        while (iss >> word) {
        transform(word.begin(), word.end(), word.begin(), ::tolower);
        invertedIndex[word].insert(docId);
        }
        }

        return invertedIndex;
}

void printInvertedIndex(const map<string, set<int>> &invertedIndex) {
        cout << "Inverted Index:\n";
        for (const auto &entry : invertedIndex) {
        cout << entry.first << ": ";
        for (int docId : entry.second) {
        cout << docId << " ";
        }
        cout << "\n";
        }
}

set<int> retrieveDocuments(const map<string, set<int>> &invertedIndex, const string
&query) {
        string lowerQuery = query;
        transform(lowerQuery.begin(), lowerQuery.end(), lowerQuery.begin(),
::tolower);

        auto it = invertedIndex.find(lowerQuery);
        if (it != invertedIndex.end()) {
        return it->second;
        } else {
        return set<int>();
        }
}

void printQueryResults(const set<int> &results, const vector<string> &documents) {
        cout << "\nQuery Results:\n";
        if (!results.empty()) {
        for (int docId : results) {
        cout << "Document " << docId << ": " << documents[docId] << "\n";
        }
        } else {
        cout << "No documents found.\n";
        }
}
```

```
int main() {
	string filename = "documents.txt";
	vector<string> documents = readDocuments(filename);

	if (documents.empty()) {
	cout << "No documents to process. Exiting.\n";
	return 1;
	}

	printDocuments(documents);

	map<string, set<int>> invertedIndex = buildInvertedIndex(documents);
	printInvertedIndex(invertedIndex);

	string query;
	cout << "Enter your query: ";
	getline(cin, query);

	set<int> results = retrieveDocuments(invertedIndex, query);
	printQueryResults(results, documents);

	return 0;
}
```

Output:

```
student@student:~/ISR$ g++ ass3.cpp
student@student:~/ISR$ ./a.out
```

Documents:
Document 0: Hello world
Document 1: Hello from the other side
Document 2: Welcome to the world of programming

Inverted Index:
from: 1
hello: 0 1
of: 2
other: 1
programming: 2
side: 1

the: 1 2
to: 2
welcome: 2
world: 0 2

Enter your query: world

Query Results:
Document 0: Hello world
Document 2: Welcome to the world of programming