Practical no 9

```
#include <iostream>
#include <string>
#include <map>
#include <limits>
using namespace std;

class Dictionary {
public:
    void addKeyword(const string& keyword, const string& meaning) {
        data[keyword] = meaning;
    }

    void deleteKeyword(const string& keyword) {
        data.erase(keyword);
    }

    void updateKeyword(const string& keyword, const string& newMeaning) {
        data[keyword] = newMeaning;
    }

    void displayAscending() const {
        map<string, string>::const_iterator it;
        for (it = data.begin(); it != data.end(); ++it) {
            cout << it->first << ": " << it->second << endl;
        }
    }

    void displayDescending() const {
        map<string, string>::const_reverse_iterator it;
        for (it = data.rbegin(); it != data.rend(); ++it) {
            cout << it->first << ": " << it->second << endl;
        }
    }

    int getMaxComparisons(const string& keyword) const {
        int count = 0;
        map<string, string>::const_iterator it;
        for (it = data.begin(); it != data.end(); ++it) {
            ++count;
            if (it->first == keyword) return count;
        }
        return count; // Keyword not found
```

```
    }

private:
    map<string, string> data;
};

// Helper functions
void printMenu() {
    cout << "\n--- Dictionary Menu ---\n"
         << "1. Add keyword\n"
         << "2. Delete keyword\n"
         << "3. Update keyword\n"
         << "4. Display ascending\n"
         << "5. Display descending\n"
         << "6. Find max comparisons\n"
         << "7. Quit\n"
         << "Enter your choice: ";
}

void clearInputBuffer() {
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
}

string inputLine(const string& prompt) {
    cout << prompt;
    string input;
    getline(cin, input);
    return input;
}

int main() {
    Dictionary dict;
    int choice;
    do {
        printMenu();
        cin >> choice;
        if (cin.fail()) {
            cin.clear();
            clearInputBuffer();
            cout << "Invalid input. Please enter a number.\n";
            continue;
        }
        clearInputBuffer(); // Clean up newline
        string keyword, meaning;
```

```
        switch (choice) {
            case 1:
                keyword = inputLine("Enter keyword: ");
                meaning = inputLine("Enter meaning: ");
                dict.addKeyword(keyword, meaning);
                break;
            case 2:
                keyword = inputLine("Enter keyword to delete: ");
                dict.deleteKeyword(keyword);
                break;
            case 3:
                keyword = inputLine("Enter keyword to update: ");
                meaning = inputLine("Enter new meaning: ");
                dict.updateKeyword(keyword, meaning);
                break;
            case 4:
                cout << "\n--- Dictionary (Ascending Order) ---\n";
                dict.displayAscending();
                break;
            case 5:
                cout << "\n--- Dictionary (Descending Order) ---\n";
                dict.displayDescending();
                break;
            case 6:
                keyword = inputLine("Enter keyword to search: ");
                cout << "Max comparisons for '" << keyword << "': "
                     << dict.getMaxComparisons(keyword) << endl;
                break;
            case 7:
                cout << "Exiting... Goodbye!\n";
                break;
            default:
                cout << "Invalid choice. Try again.\n";
        }
    } while (choice != 7);
    return 0;
}
```