```cpp
#include <fstream>
#include <string>
#include <map>
#include <sstream>
#include <iostream>
#include <deque>


using namespace std;

using tabla = map < string, int >;

void actualizaciones(tabla const& antiguo, tabla const& nuevo) {

    map <string, deque<string>> solucion;

    auto ita = antiguo.cbegin();
    auto itn = nuevo.cbegin();

    while (ita != antiguo.cend() && itn != nuevo.cend()) {

        if (ita->first == itn->first) {

            if (ita->second != itn->second) {
                solucion["*"].push_back(ita->first);
            }

            ita++;
            itn++;
        }
        else if (ita->first < itn->first) {

            solucion["-"].push_back(ita->first);
            ita++;
        }
        else if (ita->first > itn->first) {

            solucion["+"].push_back(itn->first);
            itn++;
        }
    }

    while (ita != antiguo.cend()) {

        solucion["-"].push_back(ita->first);
        ita++;
    }

    while (itn != nuevo.cend()) {

        solucion["+"].push_back(itn->first);
        itn++;
    }
```

```
    if (solucion.count("+") == 0 && solucion.count("-") == 0 && solucion.count("*") == 0)
{

        cout << "Sin cambios\n";
    }
    else {

        auto mostrar = solucion.find("+");

        if (mostrar != solucion.cend()) {

            cout << mostrar->first;

            deque<string> m = mostrar->second;

            while (!m.empty()) {

                cout << " " << m.front();
                m.pop_front();
            }

            cout << endl;
        }

        mostrar = solucion.find("-");

        if (mostrar != solucion.cend()) {

            cout << mostrar->first;

            deque<string> m = mostrar->second;

            while (!m.empty()) {

                cout << " " << m.front();
                m.pop_front();
            }

            cout << endl;
        }

        mostrar = solucion.find("*");

        if (mostrar != solucion.cend()) {

            cout << mostrar->first;

            deque<string> m = mostrar->second;

            while (!m.empty()) {

                cout << " " << m.front();
                m.pop_front();
            }
```

```
            cout << endl;
        }
    }

    cout << "---" << endl;
}


void resuelveCaso() {

    tabla antiguo;
    tabla nuevo;
    pair<string , int> elem;

    string diccionario, p;
    int n;

    getline(cin, diccionario);

    stringstream ss(diccionario);

    while (ss >> p) {

        ss >> n;

        elem.first = p;
        elem.second = n;
        antiguo.insert(elem);
    }

    getline(cin, diccionario);

    stringstream ss2(diccionario);

    while (ss2 >> p) {

        ss2 >> n;

        elem.first = p;
        elem.second = n;
        nuevo.insert(elem);
    }

    actualizaciones(antiguo, nuevo);
}

int main() {
# ifndef DOMJUDGE
    std::ifstream in("datos.txt");
    auto cinbuf = std::cin.rdbuf(in.rdbuf()); // save old buf and redirect std :: cin to
casos .txt
# endif
```

```
    int casos;

    cin >> casos;

    cin.ignore();

    for (int i = 0; i < casos; i++) {
        resuelveCaso();
    }

# ifndef DOMJUDGE // para dejar todo como estaba al principio
    std::cin.rdbuf(cinbuf);
    system(" PAUSE ");
# endif
    return 0;
}
```