```cpp
#include <iostream>
#include <fstream>
#include <unordered_map>
#include <map>
#include <list>
#include <vector>

using namespace std;

class oficinaEmpleo {

private:

    unordered_map<string, list<string>> empleosL;
    unordered_map<string, map<string, list<string>::iterator>> personasL;

public:

    oficinaEmpleo() {}

    void altaOficina(string nombre, string empleo) { //O(log n)

        if (personasL[nombre].count(empleo) == 0) {

            auto it = empleosL[empleo].insert(empleosL[empleo].end(), nombre); //O(1)
            personasL[nombre].insert({ empleo, it }); //O(log n)
        }
    }

    string ofertaEmpleo(string empleo) { //O(n)

        auto it = empleosL.find(empleo); //O(1)

        if (it == empleosL.cend()) {
            throw domain_error("No existen personas apuntadas a este empleo"); //O(1)
        }

        string per;

        per = it->second.front(); //O(1)

        for (auto const& a : personasL[per]) { //O(n)

            empleosL[a.first].erase(a.second); //O(1)

            if (empleosL[a.first].empty()) empleosL.erase(a.first); //O(1)
        }

        personasL.erase(per); //O(1)

        return per;
    }
```

```
    vector<string> listadoEmpleos(string persona) { //O(n)

        auto it = personasL.find(persona); //O(1)

        if (it == personasL.cend()) {
            throw domain_error("Persona inexistente"); //O(1)
        }

        int i = 0;
        vector<string> sol;

        for (auto const& a : it->second) { //O(n)

            sol.push_back(a.first); //O(1)
            i++;
        }

        return sol;
    }
};


bool resuelveCaso() {

    string comando, p, e;

    cin >> comando;

    if (!cin) {
        return false;
    }
    else {

        oficinaEmpleo oficina;

        while (comando != "FIN") {
            try {
                if (comando == "altaOficina") {
                    cin >> p >> e;
                    oficina.altaOficina(p, e);
                }
                else if (comando == "ofertaEmpleo") {
                    cin >> e;
                    p = oficina.ofertaEmpleo(e);
                    cout << e << ": " << p << endl;
                }
                else if (comando == "listadoEmpleos") {
                    cin >> p;
                    vector<string> sol;

                    sol = oficina.listadoEmpleos(p);

                    cout << p << ":";
```

```
                    for (int i = 0; i < sol.size(); i++) {
                        cout << " " << sol[i];
                    }

                    cout << endl;
                }
            }
            catch (domain_error& e) {
                cout << "ERROR: " << e.what() << endl;
            }

            cin >> comando;
        }

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

        return true;
    }
}

int main() {
    // ajustes para que cin extraiga directamente de un fichero
#ifndef DOMJUDGE
    std::ifstream in("datos.txt");
    auto cinbuf = std::cin.rdbuf(in.rdbuf());
#endif

    while (resuelveCaso());

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