```cpp
#include <iostream>
#include <fstream>
#include <iomanip>

#include <string>

using namespace std;

const int MAX_PEDIDOS = 3;
const int MAX_CARTAS = 10;
const int MAX_JUGUETES = 10;
const int MAX_NOMBRES = 10;

struct tCartas {
    string nombre;
    string ciudad;
    bool preparado; // cargados todos los juguetes en el trineo.
    bool pendiente; // pendiente de cargar en el trineo.
    string listaJuguetes[MAX_PEDIDOS];
};

struct tListaCartas {
    tCartas listaCartas[MAX_CARTAS];
    int cont;
};

struct tJuguetes {
    string ident;
    double peso;
    int cantidad;
};

struct tListaJuguetes {
    tJuguetes juguetes[MAX_JUGUETES];
    int cont;
};

bool cargarJuguetes(tListaJuguetes& lj) {
    ifstream fJuguetes;
    bool ok = true;
    fJuguetes.open("juguetes.txt");
    if (fJuguetes.is_open()) {
        int i = 0;
        string id;
        fJuguetes >> id;
        while (i < MAX_JUGUETES && id != "XXX") {
            lj.juguetes[i].ident = id;
            fJuguetes >> lj.juguetes[i].peso >> lj.juguetes[i].cantidad;
            fJuguetes >> id;
            i++;
        }
        lj.cont = i;
        fJuguetes.close();
    }
    else ok = false;
    return ok;
}

bool cargarCartas(tListaCartas& l) {
    ifstream fcartas;
    bool ok = true;
    fcartas.open("cartas.txt");
    if (fcartas.is_open()) {
        l.cont = 0;
        string aux;
        fcartas >> aux;
        while (l.cont < MAX_CARTAS && aux != "XXX") {
            l.listaCartas[l.cont].nombre = aux;
            fcartas >> l.listaCartas[l.cont].ciudad;
            for (int j = 0; j < MAX_PEDIDOS; ++j) {
                fcartas >> l.listaCartas[l.cont].listaJuguetes[j];
            }
            l.listaCartas[l.cont].preparado = false;
            l.listaCartas[l.cont].pendiente = false;
            l.cont++;
            fcartas >> aux;
        }
        fcartas.close();
    }
    else ok = false;
    return ok;
}

void listarJuguetes(const tListaJuguetes& lj) {
    cout << "Listado de los juguetes que hay en la fabrica\n";
    cout << setw(MAX_NOMBRES) << left << "Nombre" << setw(MAX_NOMBRES) << "Peso" << setw(MAX_NOMBRES) << "Cant." << '\n';
    for (int i = 0; i < lj.cont; ++i) {
        cout << setw(MAX_NOMBRES) << left << lj.juguetes[i].ident;
        cout << setw(MAX_NOMBRES) << fixed << setprecision(2) << lj.juguetes[i].peso;
        cout << setw(MAX_NOMBRES) << lj.juguetes[i].cantidad;
        cout << '\n';
    }
}

int buscarJuguete(const tListaJuguetes& lj, const string& nombre) {
    int i = 0;
    while (i < lj.cont && lj.juguetes[i].ident != nombre) i++;
    return i;
}

bool comprobarPedido(string pedido[MAX_PEDIDOS], const tListaJuguetes& lj) {
    int i = 0;
    bool seguir = true;
    while (i < MAX_PEDIDOS && seguir) {
        int pos = buscarJuguete(lj, pedido[i]);
        if (pos == lj.cont || lj.juguetes[pos].cantidad <= 0) {
            seguir = false;
        }
        else {
            i++;
        };
    }
    return seguir;
}

double trineoDestino(tListaCartas& l, const string& ciudad, tListaJuguetes& lj) {
    cout << "\nCargando el trineo con destino " << ciudad << '\n';
    double sumaPeso = 0;
    for (int i = 0; i < l.cont;++i) {
        if (l.listaCartas[i].ciudad == ciudad) {
            if (comprobarPedido(l.listaCartas[i].listaJuguetes, lj)) {
                cout << "Carta : " << l.listaCartas[i].nombre << "  Juguetes :";
                for (int j = 0; j < MAX_PEDIDOS; ++j) {
                    cout << ' ' << l.listaCartas[i].listaJuguetes[j];
                    int pos = buscarJuguete(lj, l.listaCartas[i].listaJuguetes[j]);
                    lj.juguetes[pos].cantidad--;
                    sumaPeso = sumaPeso +  lj.juguetes[pos].peso;
                }
                cout << '\n';
                // los juguetes de la carta existen todos
                l.listaCartas[i].preparado = true;
            }
            else l.listaCartas[i].pendiente = true;
        }
    }
    return sumaPeso;
}

bool buscarCiudad(const tListaCartas& l, string& ciudad) {
    int i = 0;
    bool encontrado = false;
    while (i < l.cont && !encontrado) {
        if (!l.listaCartas[i].preparado && !l.listaCartas[i].pendiente){
            encontrado = true;
        }
        else {
            i++;
        }
    }
    if (encontrado) ciudad = l.listaCartas[i].ciudad;
    return encontrado;
}

void listarPendientes(tListaCartas l) {
    cout << "\nListado de cartas sin cargar en los trineos \n\n";
    cout << setw(10) << left << "Nombre" << left << setw(10) << "Ciudad" << left << setw(30) << "Juguetes" << '\n';
    for (int i = 0; i < l.cont; i++) {
        if (l.listaCartas[i].pendiente) {
            cout << left << setw(MAX_NOMBRES) << l.listaCartas[i].nombre;
            cout << left << setw(MAX_NOMBRES) << l.listaCartas[i].ciudad;
            for (int j = 0; j < MAX_PEDIDOS; j++)
                cout << left << setw(MAX_NOMBRES) << l.listaCartas[i].listaJuguetes[j];
            cout << '\n';
        }
    }
}

void cargarTrineos(tListaCartas& l, tListaJuguetes& lj) {
    cout << "\nComenzando a cargar los trineos...\n";
    // Buscar la primera ciudad que no se ha cargado
    string ciudad;
    bool fin = buscarCiudad(l, ciudad);
    while (fin) {
        double peso = trineoDestino(l, ciudad, lj);
        cout << "Peso del trineo con destino " << ciudad << " : " << peso << '\n';
        fin = buscarCiudad(l, ciudad);
    }
    cout << "Finalizada la carga de todos los trineos\n";
}

int main() {
    tListaJuguetes lj;
    bool ok = cargarJuguetes(lj);
    if (!ok) std::cout << "El fichero de juguetes no se ha abierto correctamente\n";
    else {
        tListaCartas l;
        ok = cargarCartas(l);
        if (!ok) std::cout << "El fichero de cartas no se ha abierto correctamente\n";
        else {
            cout << "Bienvenido a la fabrica de juguetes de Papa Noel\n";
            cout << setw(48) << setfill('-') << '-' << '\n' << setfill(' ');

            // Listar los juguetes de la fabrica
            listarJuguetes(lj);

            // Cargar los trineos
            cargarTrineos(l, lj);

            // Listar los juguetes que quedan pendientes de enviar
            listarPendientes(l);
        }
    }
    return 0;
}
```