```
#ifndef LISTAAGENTES_H_INCLUDED
#define LISTAAGENTES_H_INCLUDED

#include <string>

#include "nodoagente.h"
#include "agente.h"
#include "listexception.h"

class ListaAgentes{
private:
    NodoAgente *ultimoInsertado;
    NodoAgente *primerInsertado;
    NodoAgente *auxiliar1;
    NodoAgente *auxiliar2;

    void intercambiar(NodoAgente*,NodoAgente*);
public:
    ListaAgentes();

    bool isEmpty();
    void insertar(const Agente&);
    void eliminar(const Agente&);
    NodoAgente* primerNodo();
    NodoAgente* ultimoNodo();
    NodoAgente* anterior(NodoAgente*);
    NodoAgente* siguiente(NodoAgente*);
    NodoAgente* localiza(const Agente&);
    void ordena();
    std::string recupera(const Agente&);
    std::string toString();
    void eliminarTodo();

};

#endif // LISTAAGENTES_H_INCLUDED


#include "listaagentes.h"

using namespace std;

void ListaAgentes::intercambiar(NodoAgente*a, NodoAgente*b) {
    Agente aux(a->getAgente());
    a->setAgente(b->getAgente());
    b->setAgente(aux);
}

ListaAgentes::ListaAgentes() : ultimoInsertado(nullptr), primerInsertado(nullptr), auxiliar1(nullptr),
```

```
auxiliar2(nullptr) {

}

bool ListaAgentes::isEmpty() {
    return ultimoInsertado == nullptr;
}

void ListaAgentes::insertar(const Agente&a) {
    NodoAgente* nuevo_nodo = new NodoAgente();
    if (primerInsertado == nullptr) {
        primerInsertado = nuevo_nodo;
        nuevo_nodo->setAgente(a);
        nuevo_nodo->setSiguiente(ultimoInsertado);
        ultimoInsertado = nuevo_nodo;
    } else {
        nuevo_nodo->setAgente(a);
        nuevo_nodo->setSiguiente(ultimoInsertado);
        ultimoInsertado->setAnterior(nuevo_nodo);
        ultimoInsertado = nuevo_nodo;
    }

}

void ListaAgentes::eliminar(const Agente&a) {
    if(isEmpty() == true) {
        throw ListException("No hay datos");
    }
    auxiliar1 = ultimoInsertado;
    while(auxiliar1 != nullptr) {
        if(auxiliar1->getAgente() == a) {
            auxiliar2 = auxiliar1->getAnterior();
            auxiliar2->setSiguiente(auxiliar1->getSiguiente());
            delete auxiliar1;
        }
        auxiliar1=auxiliar1->getSiguiente();
    }
}

NodoAgente* ListaAgentes::primerNodo() {
    if(isEmpty() == true) {
        throw ListException("No hay datos");
    }
    return primerInsertado;
}

NodoAgente* ListaAgentes::ultimoNodo() {
    if(isEmpty() == true) {
```

```
        throw ListException("No hay datos");
    }
    return ultimoInsertado;
}

NodoAgente* ListaAgentes::anterior(NodoAgente*a) {
    if(isEmpty() == true) {
        throw ListException("No hay datos");
    }
    return a->getAnterior();
}

NodoAgente* ListaAgentes::siguiente(NodoAgente*a) {
    if(isEmpty() == true) {
        throw ListException("No hay datos");
    }
    return a->getSiguiente();
}

NodoAgente* ListaAgentes::localiza(const Agente&a) {
    if(isEmpty() == true) {
        throw ListException("No hay datos");
    }
    auxiliar1 = ultimoInsertado;
    while(auxiliar1 != nullptr) {
        if(auxiliar1->getAgente() == a) {
            return auxiliar1;
        }
        auxiliar1= auxiliar1->getSiguiente();
    }
    throw ListException("No encontrado");
}

void ListaAgentes::ordena() {
    if(isEmpty() == true) {
        throw ListException("No hay datos a ordenar");
    }
    NodoAgente *auxiliar3;
    auxiliar1 = primerInsertado;
    auxiliar2 = ultimoInsertado;
    bool bandera;
    do {
        bandera = false;
        auxiliar2 = ultimoInsertado;
        while(auxiliar2 != auxiliar1) {
            auxiliar3 = auxiliar2->getSiguiente();
            if(auxiliar2->getAgente() > auxiliar3->getAgente()) {
                intercambiar(auxiliar2,auxiliar3);
```

```
            bandera = true;
        }
        auxiliar2 = auxiliar2->getSiguiente();
    }
    auxiliar1 = auxiliar1->getAnterior();
  } while(bandera);
}

string ListaAgentes::recupera(const Agente&a) {
  auxiliar1 = ultimoInsertado;
  while(auxiliar1->getSiguiente() != nullptr) {
    if(auxiliar1->getAgente() == a) {
      Agente aux(auxiliar1->getAgente());
      return aux.toString();
    }
    auxiliar1 = auxiliar1->getSiguiente();
  }
  throw ListException("No encontrado");
}

string ListaAgentes::toString() {
  if(isEmpty() == true) {
    throw ListException("No hay datos");
  }
  string resultado;
  auxiliar1 = ultimoInsertado;
  while(auxiliar1 != nullptr) {
    resultado += auxiliar1->toString();
    resultado += "\n";
    auxiliar1 = auxiliar1->getSiguiente();
  }
  return resultado;
}

void ListaAgentes::eliminarTodo() {
  if(isEmpty() == true) {
    throw ListException("No hay datos");
  }
  auxiliar1 = ultimoInsertado;
  while(auxiliar1 != nullptr) {
    auxiliar2 = auxiliar1;
    auxiliar1 = auxiliar1->getSiguiente();
    delete auxiliar2;
  }
  ultimoInsertado = nullptr;
  primerInsertado = nullptr;
```