```cpp
#include <iostream>
#include <locale>

using namespace std;

template<class T>
class LSLSE;

template<class T>
class node{
private:
    T data;
    node<T>* sig;
public:
    node():sig(nullptr){};
    friend class LSLSE<T>;
};
///********************

class SocioClub{
private:
    int nSocio;
    char nomSocio;
    char domicilio;
    int ingreso;
public:
    SocioClub(){};

    bool operator == (const SocioClub& e) const {
        return nSocio == e.nSocio;
    }
    bool operator != (const SocioClub& e) const {
        return nSocio != e.nSocio;
    }
    bool operator < (const SocioClub& e) const {
        return nSocio < e.nSocio;
    }
    bool operator <= (const SocioClub& e) const {
        return nSocio <= e.nSocio;
    }
    bool operator > (const SocioClub& e) const {
        return nSocio > e.nSocio;
    }
    bool operator >= (const SocioClub& e) const{
        return nSocio >= e.nSocio;
    }
    friend std::istream& operator >> (std::istream& g, SocioClub& o){
        cout << endl << endl;
        cout <<" Numero de socio: ";
        g>>o.nSocio;
        cout << " Nombre: ";
        g>>o.nomSocio;
        cout << " Domicilio: ";
        g>>o.domicilio;
        cout << " Año de ingreso: ";
        g>>o.ingreso;
        cout << endl << endl;
        return g;

    }
    friend std::ostream& operator << (std::ostream& g, const SocioClub& o){
        g << " Numero de socio: " << o.nSocio << endl;
        g << " Nombre: " << o.nomSocio << endl;
        g << " Domicilio: " << o.domicilio << endl;
        g << " Año de ingreso: " << o.ingreso << endl;
        return g;
```

```
    }

};
///*******************
template<class T>
class LSLSE{
private:
    SocioClub datos;
    node<T>* lista;
    int ult;
    int cont;
public:
    LSLSE():lista(nullptr){};
    bool vacia()const;
    bool llena();
    node<T>* ultimo()const;
    node<T>* primero()const;
    node<T>* anterior(node<T>* pos)const;
    void insertar(node<T>* pos, T elem);
    bool eliminar(node<T>* pos);
    void imprimir()const;
    node<T>* busca (T e)const;
    void agrega(T e);
};
template<class T>
 node<T>* LSLSE<T>::busca(const T e)const{
    node<T>* aux = lista;
    while(aux != nullptr){
        if(aux->data == e){
            return aux;
        }
        aux = aux->sig;
    }
    return nullptr;
}
template<class T>
bool LSLSE<T>::llena(){
    return cont == 99;
}
template<class T>
void LSLSE<T>::agrega(T e){
    node<T>* aux = new node<T>;
    aux->data = e;
    aux->sig = lista;
    lista = aux;
}
template<class T>
void LSLSE<T>::imprimir()const{
    node<T>* aux=lista;
    while(aux!=nullptr){
        cout << endl;
        std::cout<<aux->data<<" "<< endl;
        aux=aux->sig;
    }
}
template<class T>
bool LSLSE<T>::eliminar(node<T>* pos){
    if(vacia() || pos==nullptr){
        return false;
    }
    if(pos==lista){
        lista=lista->sig;
    }
    else{
        anterior(pos)->sig=pos->sig;
    }
```

```
    delete pos;
    return true;
}

template<class T>
void LSLSE<T>::insertar(node<T>* pos, T elem){
    node<T>* aux= new node<T>;
    aux->data=elem;
    if(pos==nullptr){
        aux->sig=lista;
        lista=aux;
    }
    else{
        aux->sig=pos->sig;
        pos->sig=aux;
    }
}

template<class T>
node<T>* LSLSE<T>::anterior(node<T>* pos)const{
    if(vacia() || pos==nullptr){
        return nullptr;
    }
    node<T>* aux=lista;
    while(aux!=nullptr && aux->sig!=pos){
        aux=aux->sig;
    }
    return aux;
}

template<class T>
node<T>* LSLSE<T>::primero()const{
    if(vacia()){
        return nullptr;
    }
    return lista;
}

template<class T>
node<T>* LSLSE<T>::ultimo()const{
    if(vacia()){
        return nullptr;
    }
    node<T>* aux=lista;
    while(aux->sig!=nullptr){
        aux=aux->sig;
    }
    return aux;
}

template<class T>
bool LSLSE<T>::vacia()const{
    if(lista==nullptr)
        return true;
    return false;
}

int main()
{
    setlocale(LC_CTYPE,"Spanish");
    LSLSE<SocioClub> miLista;
    SocioClub miSocio;
    int opc;
    int opcc;

    do{
```

```
cout << "|****LSLSE****|" << endl;
cout << " 1. Registrar un nuevo socio" << endl;
cout << " 2. Dar de baja un socio" << endl;
cout << " 3. Buscar" << endl;
cout << " 4. Imprimir" << endl;
cout << " 5. Salir" << endl;
cout << " Elige una opcion: ";
cin >> opc;

switch(opc){
    case 1: cin >> miSocio;
            miLista.agrega(miSocio);
        break;
    case 2: cout << endl;
            cout << " Ingresa el socio a eliminar... " << endl;
            cin >> miSocio;
            if(!miLista.busca(miSocio)){
                cout << " El socio no existe " << endl;
            }
            else{
                miLista.eliminar(miLista.busca(miSocio));
                cout << " Se ha eliminado exitosamente..." << endl << endl;
            }

        break;
    case 3: cout << endl;
            cout << " 1. Por nombre " << endl;
            cout << " 2. Por domicilio " << endl;
            cout << " Elige como quieres buscar: ";
            cin >> opcc;

            switch(opcc){
                case 1: cout << " Escribe el nombre del socio: ";
                        cin >> miSocio;
                        if(!miLista.busca(miSocio)){
                            cout << " Socio inexistente..." << endl << endl;
                        }
                        else{
                            cout << "**********************\n";
                            cout << miSocio;
                            cout << "**********************\n";
                            cout << endl;
                        }
                    break;
                case 2: cout << " Escribe el domicilio del socio: ";
                        cin >> miSocio;
                        if(!miLista.busca(miSocio)){
                            cout << " Socio inexistente..." << endl << endl;
                        }
                        else{
                            cout << "**********************\n";
                            cout << miSocio;
                            cout << "**********************\n";
                            cout << endl;
                        }
                    break;
                default: cout << endl;
                        cout << " Opcion invalida... " << endl << endl;
                    break;
            }
        break;
    case 4: cout << endl;
            miLista.imprimir();
        break;
    case 5: cout << endl;
            cout << " Saliendo..." << endl << endl;
```

```
            break;
        default:cout << endl;
                cout << " Opcion invalida... " << endl << endl;
            break;
    }


    }while(opc!=5);

    return 0;
}
```