```cpp
#include <iostream>
#include <string>
#include <stdlib.h>
#include <cstring>

using namespace std;

class Empleado {
private:
    int claveEmpleado;
    char nombre[100];
    char domicilio[100];
    float sueldo;
    char reportaA[100];

public:
    // Constructor
    Empleado(){};
    Empleado(int clave, const char* nombre, const char* domicilio, float sueldo, const
char* reportaA) {
        claveEmpleado = clave;
        strcpy(this->nombre, nombre);
        strcpy(this->domicilio, domicilio);
        this->sueldo = sueldo;
        strcpy(this->reportaA, reportaA);
    }

    // Método para imprimir los datos del empleado
    /*void imprime() {
        cout << "Clave de Empleado: " << claveEmpleado << endl;
        cout << "Nombre: " << nombre << endl;
        cout << "Domicilio: " << domicilio << endl;
        cout << "Sueldo: " << sueldo << endl;
        cout << "Reporta a: " << reportaA << endl;
    }*/

    // Método para cambiar el domicilio
    void cambiaDomic(const char* nuevoDomicilio) {
        strcpy(domicilio, nuevoDomicilio);
    }

    // Método para cambiar el supervisor al que reporta
    void cambiaReportaA(const char* nuevoSupervisor) {
        strcpy(reportaA, nuevoSupervisor);
    }

    // Método para actualizar el sueldo
    void actualSueldo(float nuevoSueldo) {
        sueldo = nuevoSueldo;
    }

    bool operator == (const Empleado& e) const {
        return nombre == e.nombre;
    }
    bool operator != (const Empleado& e) const {
        return nombre != e.nombre;
    }
    bool operator < (const Empleado& e) const {
        return nombre < e.nombre;
    }
    bool operator <= (const Empleado& e) const {
        return nombre <= e.nombre;
    }
    bool operator > (const Empleado& e) const {
        return nombre > e.nombre;
    }
```

```
    bool operator >= (const Empleado& e) const{
        return nombre >= e.nombre;
    }
    friend std::istream& operator >> (std::istream& g, Empleado& o){
        cout << endl << endl;
        cout <<" Nombre: ";
        g>>o.nombre;
        cout << " Domicilio: ";
        g>>o.domicilio;
        cout << " Sueldo: ";
        g>>o.sueldo;
        cout << endl << endl;
        return g;

    }
    friend std::ostream& operator << (std::ostream& g, const Empleado& o){
        g << " Nombre: " << o.nombre << endl;
        g << " Domicilio: " << o.domicilio << endl;
        g << " Sueldo: " << o.sueldo << endl;
        return g;

    }

};

class Lista{
private:
    int tam;
    int cont;
    Empleado datos[100];

public:
    Lista(){}; //cont (-1){}
   /* std::string toString(){
        string r;
        int x(o);
        while(x <= cont){
            r+= to_string(datos[x]) + ", ";
            x++;
        }
        r+= " \n ";
        return r;

    }*/
     bool posValida (const int& p){
        return p <= 0 and p <= cont;
    }

    //posicion, elemento
    void inserta(const int& p, const Empleado& e){
        if(llena()){
            cout << " La lista esta llena..." << endl << endl;
        }
        if(p != -1 and !posValida(p)){
            cout << " Posicion invalida..." << endl << endl;
        }
        int x(cont);
        while(x > p){
            datos[x + 1] = datos[x];
            x--;
        }
        datos[p + 1] = e;
        cont ++;
    }
    //posicion
    void elimina(const int& p){
```

```
        if(!posValida(p)){
            cout << " Posicion invalida..." << endl << endl;
        }
        int x(p);
        while(x < cont){
            datos[x] = datos[x + 1];
            x++;
        }
        cont--;
    }
    void agrega(Empleado& e){
        if(llena()){
            cout << " La lista esta llena ";
        }
        else{
            datos[cont] = e;
            cont++;
        }
    }

    int busca(const Empleado& e){
        int x(0);
        while(x <= cont ){
            if(datos[x] == e){
                return x;
            }
            x++;
        }
        return -1;
    }

    void muestra(){
        if(vacia()){
            cout << " La lista esta vacia " << endl;
        }
        else{
        int x(0);
        while(x <= cont){
            cout << datos[x++] << "\n";
        }
        cout << endl;
        }
    }
    bool vacia(){
        return cont == -1;
    }
    bool llena(){
        return cont == 99;
    }
};

int main()
{
    Lista miLista;
    Empleado miEmpleado;
    int opc;
    int pos;
    int posi;
    int pp;

    do{
    cout << " |***LISTA***| " << endl << endl;
    cout << " 1. Agregar " << endl;
    cout << " 2. Buscar " << endl;
    cout << " 3. Eliminar " << endl;
    cout << " 4. Insertar " << endl;
```

```
    cout << " 5. Mostrar " << endl;
    cout << " 6. Salir " << endl;
    cout << " Elige una opcion: ";
    cin >> opc;
    cin.ignore();

    switch(opc){
        case 1: cin>>miEmpleado;
                miLista.agrega(miEmpleado);
            break;
        case 2: cout << "\n Ingrese el empleado a buscar\n ";
                cin >> miEmpleado;
                pos = miLista.busca(miEmpleado);
                cout << " El empleado con el nombre \n" << miEmpleado << " ";
                if(pos == -1){
                    cout << " No se encuentra en la lista ";
                }
                else{
                    cout << " Se encuentra en la posicion " << pos;
                    cout << endl << endl;
                }
                 cout << endl << endl;

            break;
        case 3: cout << " Escribe la posicion a eliminar: ";
                cin >> posi;
                miLista.elimina(posi);
                cout << endl << endl;

            break;
        case 4: cout << " Escribe la posicion: ";
                cin >> pp;
                cout << " Ecribe el nombre del empleado a insertar\n ";
                cin >> miEmpleado;
                miLista.inserta(pp,miEmpleado);
            break;
        case 5: miLista.muestra();
            break;
        case 6: cout << " Saliendo..." << endl << endl;
            break;
        default: cout << " Opcion invalida..." << endl << endl;
            break;
    }
    }while(opc!=6);

    return 0;
}
```