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

using namespace std;

template<class T>
class ColaDinamica;

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

class Estudiante{
private:
    std::string nombre;
    int materias;
    std::string carrera;
    float promedio;
public:
    Estudiante(){};

    bool operator == (const Estudiante& e) const {
        return nombre == e.nombre;
    }
    bool operator != (const Estudiante& e) const {
            return nombre != e.nombre;
        }
    bool operator < (const Estudiante& e) const {
            return nombre < e.nombre;
        }
    bool operator <= (const Estudiante& e) const {
            return nombre <= e.nombre;
        }
    bool operator > (const Estudiante& e) const {
            return nombre > e.nombre;
        }
    bool operator >= (const Estudiante& e) const{
            return nombre >= e.nombre;
        }
    friend  std::ostream& operator<<( std::ostream& o, Estudiante& p){
    o << " Nombre: " << p.nombre << std::endl;
    o << " Materias aprobadas: " << p.materias << std::endl;
    o << " Promedio: " << p.promedio << std::endl;
    o << " Carrera: " << p.carrera << std::endl;
    cout << endl;
    return o;
        }
    friend std::istream& operator>>( std::istream& g, Estudiante& o){
        std::cout << endl << endl;
        std::cout <<" Nombre: ";
        g>>o.nombre;
        std::cout << " Materias aprobadas: ";
        g>>o.materias;
        cout << " Promedio: ";
        g>>o.promedio;
        cout << " Carrera: ";
        g>>o.carrera;
        cout << endl << endl;
        return g;
```

```
        }
};
///*******************
template<class T>
class ColaDinamica{
private:
    Estudiante datos;
    node<T>* lista;
    int ult;
    int cont;
public:
    ColaDinamica():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;
    void imprimirPrimero()const;
    node<T>* busca (T e)const;
    void agrega(T e);
    void enqueue(T elem);
    Estudiante& dequeue();
    Estudiante& top();
};
/*template<class T>
Estudiante& ColaDinamica<T>::top(){
    if(vacia())
        cout << "\n La cola esta vacia " << endl;
    else
    return datos[ult];
}*/
template<class T>
void ColaDinamica<T>::enqueue(T elem){
    insertar(ultimo(),elem);

}
template<class T>
Estudiante& ColaDinamica<T>::dequeue(){
    eliminar(primero());
}
template<class T>
 node<T>* ColaDinamica<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 ColaDinamica<T>::llena(){
    return cont == 99;
}
template<class T>
void ColaDinamica<T>::agrega(T e){
    node<T>* aux = new node<T>;
    aux->data = e;
    aux->sig = lista;
    lista = aux;
}
```

```
template<class T>
void ColaDinamica<T>::imprimirPrimero()const{
    node<T>* aux=lista;
        cout << endl;
        std::cout<<aux->data<<" "<< endl;
}
template<class T>
void ColaDinamica<T>::imprimir()const{
    node<T>* aux=lista;
    while(aux!=nullptr){
        cout << endl;
        std::cout<<aux->data<<" "<< endl;
        aux=aux->sig;
    }
}

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

template<class T>
```

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

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


int main()
{
    ColaDinamica<Estudiante> miCola;
    Estudiante miEstudiante;
    int opc;

    do{

    cout << "|****Cola Dinamica****|" << endl;
    cout << " 1. Dar de alta una solicitud" << endl;
    cout << " 2. Elaborar una costancia" << endl;
    cout << " 3. Imprimir" << endl;
    cout << " 4. Salir" << endl;
    cout << " Elige una opcion: ";
    cin >> opc;

    switch(opc){
        case 1: cin >> miEstudiante;
                miCola.enqueue(miEstudiante);
            break;
        case 2: cout << endl;
                if(!miCola.vacia()){
                    miCola.imprimirPrimero();
                    miCola.dequeue();
                }
                else
                    cout << " La cola esta vacia..." << endl << endl;
            break;
        case 3: cout << endl;
                cout << "*************************" << endl;
                miCola.imprimir();
                cout << "*************************" << endl;
            break;
        case 4: cout << endl;
                cout << " Saliendo..." << endl << endl;
            break;
        default:cout << endl;
                cout << " Opcion invalida... " << endl << endl;
            break;
        }

    }while(opc!=4);

    return 0;
}
```