Exercici 2.8 (nivell mig). EXERCICI NO AVALUABLE

Suposem que tenim ja creades les classes AssignaturaExpedient, Estudiant i Titulació com a classe de teoria:

```
class AssignaturaExpedient
{
public:
	AssignaturaExpedient() { m_nConvocatories = 0; }
	AssignaturaExpedient(int codi, int nCredits) { m_codi = codi;
				m_nCredits = nCredits; m_nConvocatories = 0; }
	int getCodi() const { return m_codi; }
	int getCredits() const { return m_nCredits; }
	void setCodi(int codi) { m_codi=codi; }
	void setCredits(int credits) { m_nCredits = credits; }
private:
	static const int MAX_CONVOCATORIES = 3;
	int m_codi;
	int m_nCredits;
	int m_notes[MAX_CONVOCATORIES];
	int m_nConvocatories;
};
```

```
#include "Assignatura.h"
#include <forward_list>
#include <string>
using namespace std;
class Estudiant
{
public:
	Estudiant() { m_nAssigActual = 0;}
	Estudiant(const string &nom, const string &niu)
	{
		m_nom = nom;
		m_NIU = niu;
		m_nAssigActual = 0;
	}
	void mostraAssignatures();
	int getNAssigActual() { return m_nAssigActual; }
	string& getNIU() { return m_NIU; }
private:
	static const int MAX_ASSIGNATURES = 10;
	string m_nom;
	string m_NIU;
	AssignaturaExpedient m_assignatures[MAX_ASSIGNATURES];
	int m_nAssigActual;
};
```

```
#include <string>
using namespace std;
#include "Estudiant.h"

class Titulacio
{
private:
	string m_nom;
	int m_nEstudiants;
	std::forward_list<Estudiant> m_estudiants;
public:
	bool afegeixEstudiant(Estudiant &e);
	void mostraEstudiants();
};
```

Volem implementar els següents mètodes a la classe Titulació:

1. Un mètode que elimini els estudiants que no tenen cap assignatura. És a dir que encara que al se array d'assignatures tinguin espai per guardar assignatures, no hagin afegit cap.:

```
void eliminaEstudiants();
```

2. Un mètode que busqui un estudiant a la titulació i ens retorni si l'ha trobat o no. A més retorni com a paràmetre per referència un iterador a la posició a on es troba i un iterador a la posició anterior, per tal de poder modificar-lo, afegir un element a continuació o eliminar-lo:

```
bool buscaEstudiant(const string& niu,
                    std::forward_list<Estudiant>::iterator& actual,
                    std::forward_list<Estudiant>::iterator& anterior);
```

3. Com quedaria ara el mètode afegeixEstudiant que teníem fet a teoria?

```
void Titulacio::afegeixEstudiant (Estudiant &e)
{
    bool trobat = false;
   std::forward_list<Estudiant>::iterator anterior =
                                          m_estudiants.before_begin();
   std::forward_list<Estudiant>::iterator actual =
                                          m_estudiants.begin();
    while ((actual != m_estudiants.end()) && (!trobat))
    {
        Estudiant estActual = *actual;
        if (e.getNIU() < estActual.getNIU())
            trobat = true;
        else
        {
            anterior = actual;
            actual++;
        }
    }
   if (!trobat)
   { m_estudiants.insert_after(anterior, e);
     m_nEstudiants++;
   }
}
```

4. Un mètode per afegir una assignatura a un estudiant:

```
bool afegeixAssignaturaEstudiant(const string& niu,
                                 int codiAssig, int nCredits)
```

a) Primer supuseu que teniu el mètode afegeixAssignatura a la classe Estudiant.

```
bool afegeixAssignatura(int codi, int nCredits)
```

b) Ara afegiu el mètode afegeixAssignatura a Estudiant

Tingueu en compte:

- A Caronte trobareu la declaració i implementació de les classes Titulacio, Estudiant i Assignatura i que heu d'utilitzar.
- A Caronte trobareu un programa principal que us servirà per validar el funcionament correcte dels nous mètodes que afegiu a la classe titulació i Estudiant.